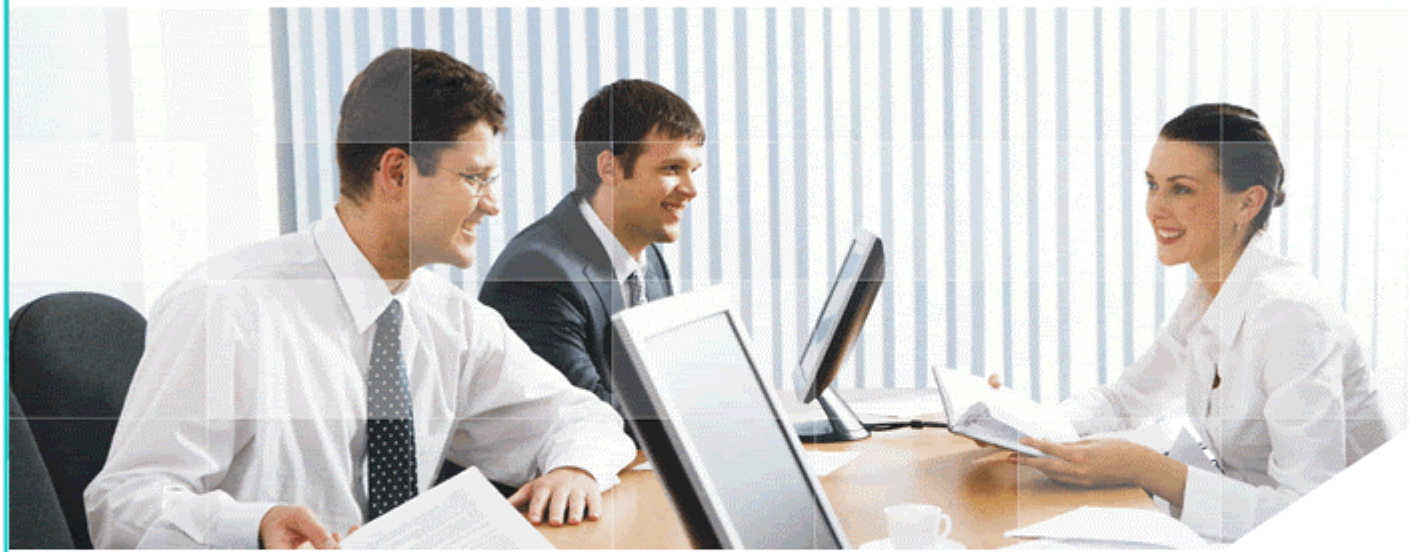
SIVECO
ROMANIA SA
Soft care contează



Informatică
Catalog AeL eContent

©2008 SIVECO Romania.
All Rights Reserved.

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru zero ore de predare.**

1. Algoritmi de sortare: Introducere
2. Noþiuni introductive: Vectori. Tablouri
3. Noþiuni introductive: Citirea unui vector
4. Noþiuni introductive: Parcurgerea unui vector
5. Noþiuni introductive: Inserþia de elemente noi în vectori
6. Sortarea prin amestecare (Shaker)
7. Bubble sort
8. Soratrea prin calcul Min/Max
9. Sortarea prin inserþie, folosind doi vectori
10. Sortare prin inserþie binarã
11. Sortare prin inserþie directã
12. Sortare prin interclasarea a doi vectori
13. Sortare prin interclasare (Merge Sort)
14. Sortarea prin interschimbare
15. Sortarea rapidã (Quick Sort)
16. Sortarea prin selecþie directã
17. Sortarea prin metoda Shell

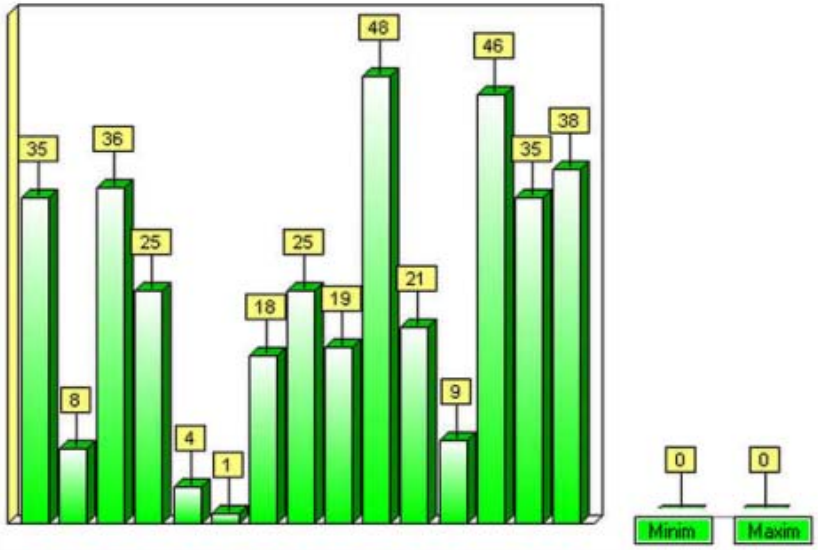
Determinare Minim / Maxim — **Algoritmi de Sortare**

Minim - cel mai mic element din vector .
Maxim - cel mai mare element din vector
- elementul minim si maxim se determina simultan .

Inspector de variabile
N =
I =
A [i] =
Min =
Max =

Pseudocod

```
Min <— A [ 1 ] ;
Max <— A [ 1 ] ;
pentru I <— 2 , N  executa
   |  daca  ( A [ I ] < Min  ) atunci
   |     |   Min <— A [ I ] ;
   |  sfarsit daca ;
   |  daca  ( A [ I ] > Max  ) atunci
   |     |   Max <— A [ I ] ;
   |  sfarsit daca ;
sfarsit pentru ;
```



Legenda
- Elementele din Vector
- Elementele din vector care sunt comparate
- Elementul Min/Max ia valoarea unui element din vector

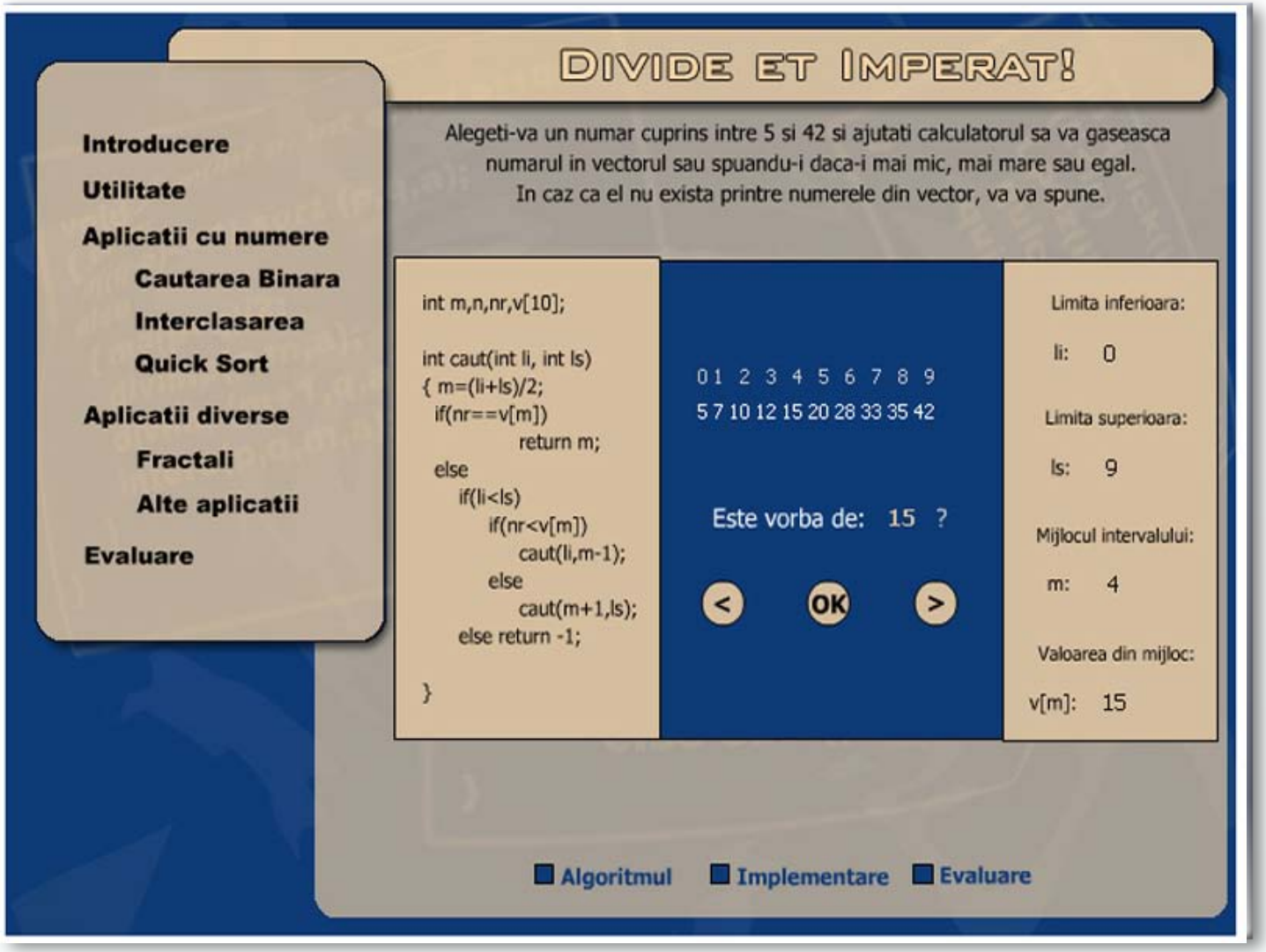
Introduce un Element
Stergere Vector
Initializare Vector
Vector Predefinit
Determinare Minim / Maxim

Vector Predefinit
Viteza de Executie

SIVECO ROMANIA SA

- **Recomandat pentru patru ore de predare.**

1. Divide et Impera



AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru doua ore de predare.**

1. Declararea ºi definirea funcþiilor
2. Apelul funcþiilor. Ordinea parametrilor
3. Apelul funcþiilor. Transmiterea parametrilor
4. Apelul funcþiilor. Exemplificare
5. Cel mai mare divizor comun
6. Cautarea secvenþialã ºi cãutarea binara
7. Test grilã de evaluare a cunoºtinþelor

Funcții Pascal

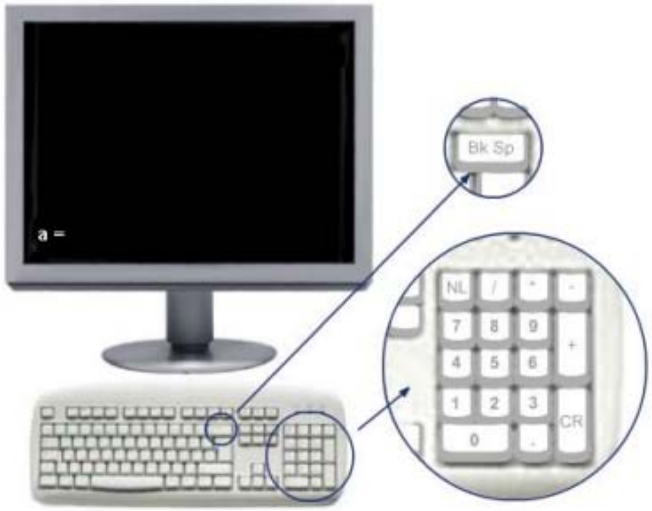
Apelul funcțiilor - Exemplificare

AeL eCONTENT
Built to teach intelli gently

Exercițiu :

Să se scre programul pentru calculul următoarei expresii :

$$E = \begin{cases} max(\, min\,(|a| - 3b,\ c - 2|d|),\ min\,(|a| + 3b,\ c + 2|d|)\,), & \text{dacă } a + b > c - d \\ min(\, max\,(|a| - 3b,\ c - 2|d|),\ max\,(|a| + 3b,\ c + 2|d|)\,), & \text{dacă } a + b \le c - d \end{cases}$$




```
program calcul_expresie;
var a, b, c, d, E : integer ;

function min (x, y : integer)  : integer ;
begin
   min := x;
   if y < x then min := y;
end;

function max (x, y : integer)  : integer ;
begin
   max := x;
   if y > x then max := y;
end;

function modul (x : integer)  : integer ;
begin
   modul := x;
   if x < 0 then modul := - x;
end;

begin
  write("a = "); readln(a);
  write("b = "); readln(b);
  write("c = "); readln(c);
  write("d = "); readln(d);
  if a+b > c-d then E := max( min( modul(a)-3*b, c-2*modul(d) ),
                              min( modul(a)+3*b, c+2*modul(d) ) )
               else E := min( max( modul(a)-3*b, c-2*modul(d) ),
                              max( modul(a)+3*b, c+2*modul(d) ) );
  writeln("E = ", E );
end.
```


SIVECO ROMANIA SA

AeL eCONTENT
Built to teach *intelli* gently

• **Recomandat pentru cinci ore de predare.**

1. Instrucţiunea if: Diagramă de sintaxă
2. Instrucţiunea if: Aplicaţie
3. Instrucţiunea switch: Diagrama de sintaxă
4. Instrucţiunea switch: Aplicaţia I
5. Instrucţiunea switch: Aplicaţia II
6. Instrucţiunea while: Diagrama de sintaxă
7. Instrucţiunea while: Aplicaţie
8. Instrucţiunea do-while: Diagrama de sintaxă
9. Instrucţiunea do-while: Aplicaţie
10. Instrucţiunea for: Diagrama de sintaxă
11. Instrucţiunea for: Aplicaţie

Instrucţiuni C++

Instrucţiunea *switch* : Aplicaţia 2

Enunţul problemei:

La un concurs de tras cu arcul, juriul acordă puncte în funcţie de culoarea zonei pe care arcaşul o nimereşte, după cum urmează: 10 puncte pentru albastru, 50 de puncte pentru roşu, 100 de puncte pentru galben şi 0 puncte pentru negru. De asemenea, în cazul în care ţinta nu este nimerită, nu se acordă nici un punct. Simulaţi acest sistem de jurizare, folosind instrucţiunea **switch**.

Puncte: 0

```
switch (culoare){
  case 1:
        puncte=10;
  break;
  case 2:
        puncte=50;
  break;
  case 3:
        puncte=100;
  break;
  default:
        puncte=0;
  break;
}
```

Numerotarea culorilor:

- → 1
- → 2
- → 3

informatică

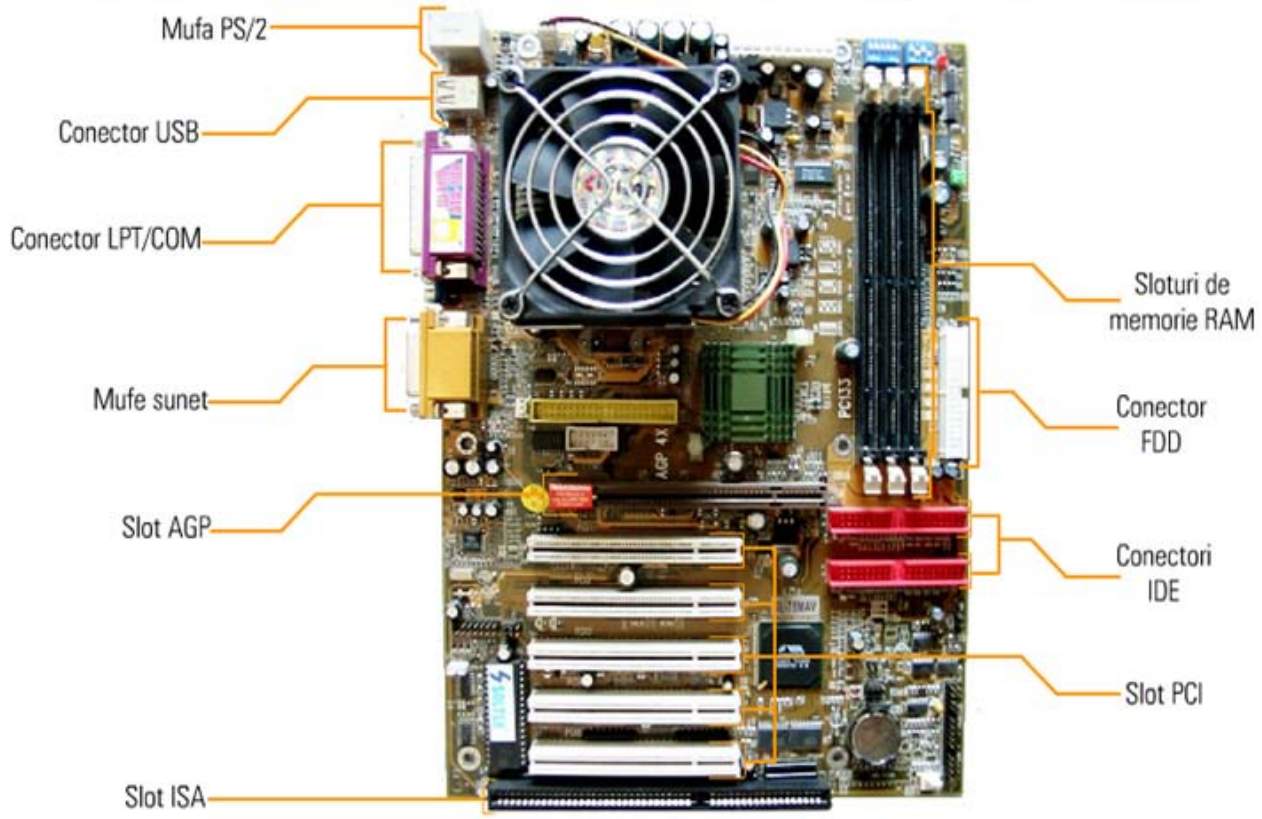
SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Arhiectura unui sistem de calcul
2. Cuplarea dispozitivelor periferice. Tipuri de interfeþe
3. Cuplarea dispozitivelor periferice. Tipuri de interfeþe: Test interactiv
4. Cuplarea interfeþelor ºi unitãþilor de memorie externã.
5. Cuplarea interfeþelor ºi unitãþilor de memorie externã: Test interactiv

§ Cuplarea interfețelor și unităților de memorie externă



Software educaţional dezvoltat de VanByte, în 2003

SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Noțiuni introductive
2. Aplicatii
3. Unitatea centrală de prelucrare
4. Miezuri feromagnetice. Înfăºurările electrice

Ce este calculatorul?

Unitatea centrală de prelucrare (UCP)

AeL eCONTENT
Built to teach intelli gently

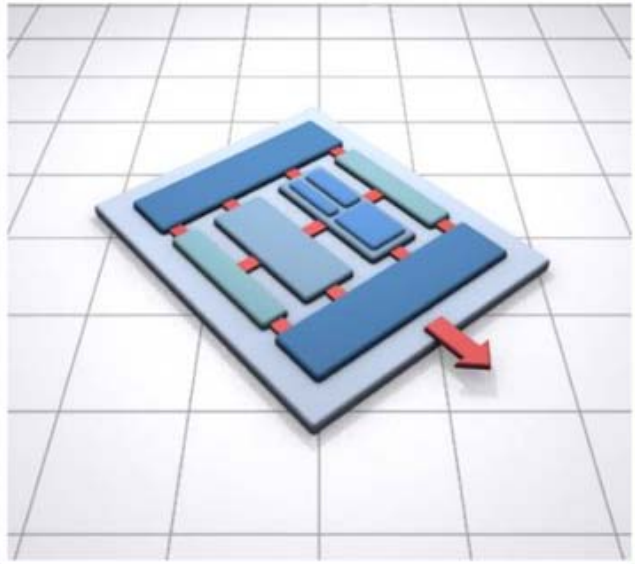
Unitatea centrală de prelucrare este implementată cu ajutorul unui **microprocesor** care este un circuit integrat ce conține milioane de tranzistori, încapsulați într-un înveliș ceramic de numai câțiva centimetri pătrați.

Microprocesorul este alcătuit dintr-un ansamblu de unități funcționale strâns legate între ele:

- **UCC (unitatea de comandă și control):** controlează activitatea tuturor dispozitivelor fizice ale calculatorului și coordonează procesul de decodificare și executare a instrucțiunilor;
- **UAL (unitatea aritmetico-logică, eng: ALU):** asigură executarea operațiilor aritmetice și logice elementare;
- **coprocesorul matematic (eng: FPU):** execută operații mai complexe (în virgulă mobilă);
- **registrele de stare și acumulator:** zone de stocare temporară a informațiilor încorporate în microprocesor, care pot fi citite și scrise foarte rapid;
- **unitatea de gestionare a intrărilor și ieșirilor:** controlează fluxurile de informații care intră și ies, având interfață directă cu memoria principală a sistemului;
- **memoria Cache (L1 și L2):** este o memorie de tip RAM cu ajutorul căreia microprocesorul poate accesa cele mai recente informații folosite mult mai rapid decât prin intermediul memoriei RAM propriu-zise;
- **magistrala:** interconectează toate aceste blocuri, asigurând o comunicare rapidă.

Schema de principiu a unui microprocesor

Indicație: Identificați unitățile funcționale ale microprocesorului poziționând mouse-ul asupra fiecărui element din imagine.



SIVECO ROMANIA SA

• **Recomandat pentru doua ore de predare.**

1. Dispozitive de intrare: Tastatura
2. Alte dispozitive periferice de intrare
3. Dispozitive periferice de ieºire: Imprimanta

Sisteme de intrare / ieşire

Dispozitive de ieşire: Monitorul

Monitorul este un dispozitiv periferic care permite vizualizarea pe un ecran a rezultatelor executării programelor.

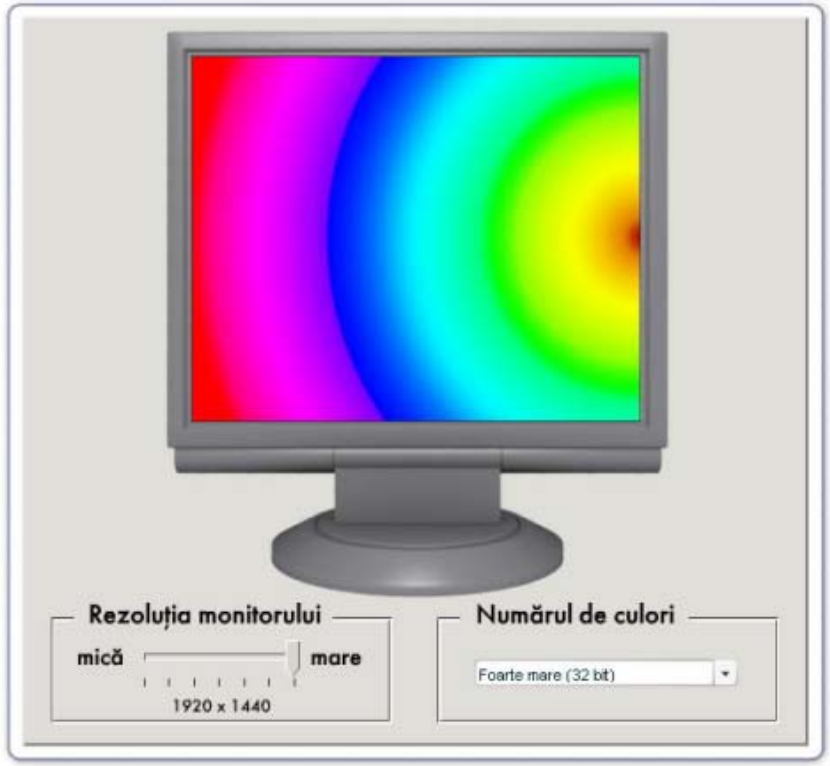
Clasificări ale monitoarelor:

- în funcţie de numărul de culori ce pot fi afişate pe ecran
- în funcţie de dimensiunea ecranului
- în funcţie de rezoluţia ecranului
- din punct de vedere tehnologic

Clasificare din punct de vedere tehnologic

- monitoare cu tub catodic (CRT = Cathode Ray Tube)
- Monitoare care au afisaj cu cristale lichide (LCD = Liquid Crystal Display)

rezoluţii mari	geometria imaginii
preţ mai mic	nu emite radiaţii
	consum energetic mic
	dimensiuni reduse



informatică

• **Recomandat pentru o ora de predare.**

1. Definirea noțiunii de sistem de operare (1)
2. Tipuri de sisteme de operare

Sisteme de operare: definiții, funcții, tipuri

Tipuri de sisteme de operare

Sistemele de operare pot fi clasificate în funcție de:

- **numărul de utilizatori ce pot folosi sistemul în același timp:**
  - **monoutilizator** , când comunicarea cu sistemul de calcul este posibilă, la un moment dat, numai pentru un singur utilizator, prin intermediul consolei sistemului de calcul;
  - **multiutilizator** , când sistemul de operare poate gestiona comunicarea concomitentă cu mai mulți utilizatori, conectați la sistemul de calcul prin intermediul echipamentelor terminale de intrare/ieșire.
- **numărul de programe ce pot fi prezente simultan în memoria calculatorului:**
  - **monotasking** , care nu asigură execuția concurentă și nici partajarea resurselor între mai multe procese. Sub controlul unui sistem de operare monotasking, la un moment dat, în sistemul de calcul se poate executa un singur program; acesta rămâne activ din momentul lansării lui în execuție și până la terminarea lui completă; cât timp este în execuție, programul are acces la toate resursele sistemului de calcul;
  - **multitasking** sunt acele sisteme de operare care asigură execuția concurentă a mai multor procese care există concomitent în sistem.

SO monoutilizator	SO multiutilizator

SO monotasking	SO multitasking

MS DOS, Windows .NET Server 2003, Windows, Mac OS, Linux, SOLARIS, OS2 Warp, FreeBSD

informatică

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Organizarea logică
2. Principiul de funcționare
2. Joc educațional

Unitatea de memorie

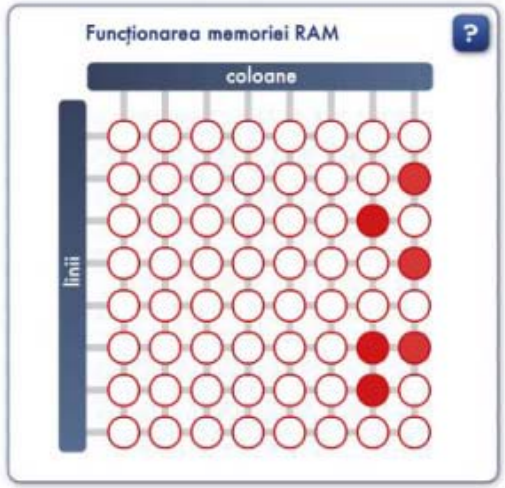
Organizarea fizică

AeL eCONTENT
Built to teach intelli gently

Tipul de memorie: RAM ROM

Memoria RAM (Random Access Memory):

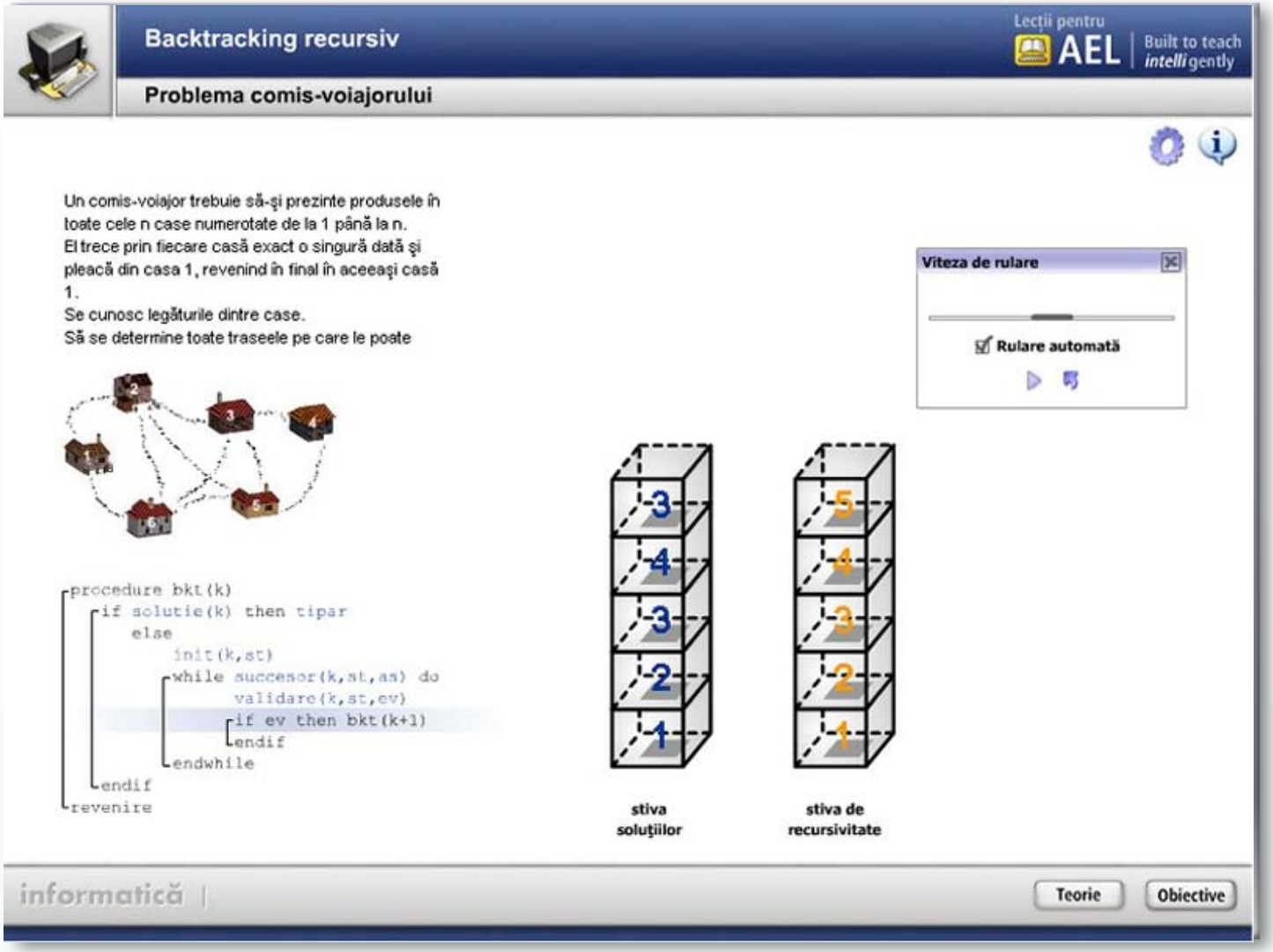
- este formată din mai multe circuite integrate, permite acces la citire și la scriere și este volatilă (conținutul ei se pierde la închiderea calculatorului)
- se clasifică în funcție de utilitate în: **memoria RAM principală, memoria RAM CACHE, memoria RAM video, memoria PRINTER Buffer, memoria CMOS**



SIVECO ROMANIA SA

• **Recomandat pentru patru ore de predare.**

1. Joc. Problema damelor
2. Exemplu practic: Aranjarea mobilei într-o casă
3. Operații pe stivă
4. Exemple de operații pe stivă
5. Prezentarea sablonului de funcții și proceduri
6. Generarea permutărilor
7. Problema generării aranjamentelor
8. Problema damelor
9. Joc: Problema comis-voiajorului
10. Problema comis-voiajorului: Rezolvare iterativă
11. Problema comis-voiajorului: Rezolvare recursiva
12. Problema comis-voiajorului: Implementare
13. Joc: Problema labirintului
14. Problema labirintului: Rezolvare iterativă
15. Problema labirintului: Implementare



• **Recomandat pentru trei ore de predare.**

1. Backtracking iterativ. Generarea aranjamentelor
2. Colorarea hărtilor
3. Colorarea hărtilor. Aplicație: Harta României
4. Backtracking recursiv. Plata unei sume de bani
5. Problema labirintului

Tehnica Backtracking

AeL eCONTENT
Built to teach intelli gently

Backtracking recursiv - Plata unei sume de bani

Se dau suma *s* și *n* tipuri de bancnote având următoarele valori: 100, 200, 500, 1000, 2000 unități monetare. Se cer toate modalitățile de plată a sumei *s* utilizând aceste bancnote.

Date de intrare :

- Suma ce trebuie plătită 2500
- Tipurile de bancnote cu care se face plata :

Reprezentarea datelor de intrare în program			
s	2500	n	0
banc[]	[ ]		
Valorile prelucrate ale datelor			
maxB[]	[ ]		
k	0	sTemp	0
sol[]	[ ]		
soluție			

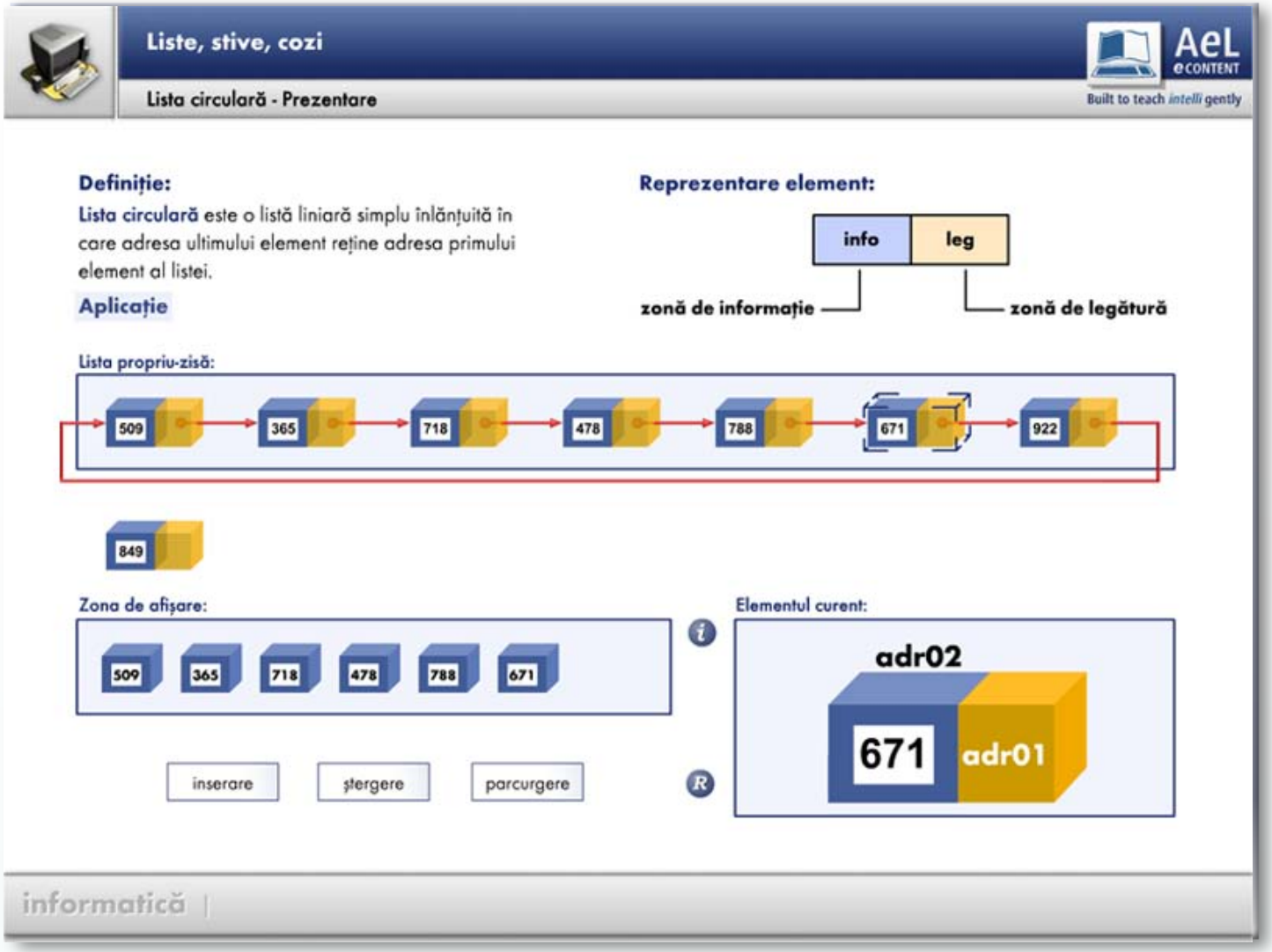
Plata unei sume de bani cu bacnote de valori date

```
# include <stdio.h>
int sol[5],banc[5],maxB[5],n,i,s;
void afis(int k)
  { printf (" solutie: \n");
    for (i=1;i<=k;i++) if (sol[i]>0)
      printf ("%d bancnote de %d\n",sol[i],banc[i]);
  }
void plata(int k, int sTemp)
  { while ( sol[k]<maxB[k] && sTemp+banc[k]<=s )
      { sol[k]++;
        if (sol[k]>0) sTemp+=banc[k];
        if (sTemp == s) afis(k);
           else if (k<n) plata(k+1,sTemp);
      }
    sol[k]=-1;
  }
void dateIntrare()
  { printf("suma de plata ="); scanf("%d",&s);
    printf("numar bacnote ="); scanf("%d",&n);
    for (i=1;i<=n;i++)
      { printf("valoarea bancnotei de tipul %d",i);
        scanf("%d",&banc[i]);
        maxB[i] = s/banc[i]; sol[i]=-1;
      }
  }
void main()
{
  dateIntrare();
  plata(1,0);
}
```

AeL eCONTENT
Built to teach *intelli* gently

• **Recomandat pentru trei ore de predare.**

1. Lista simplu înlănţuită. Prezentare
2. Lista simplu înlănţuită. Intersecţia a doua mulţimi
3. Lista simplu înlănţuită. Reuniunea a doua mulţimi
4. Lista dublu înlănţuită. Prezentare
5. Lista circulară. Prezentare
6. Liste înlanţuite. Test grilă de evaluare a cunostinţelor
7. Stiva. Prezentare
8. Stiva. Aplicaţie
9. Stiva. Test grilă de evaluare a cunoºtinţelor
10. Coada. Prezentare
11. Coada. Aplicaţie
12. Coada. Test grilă de evaluare a cunoºtinţelor



SIVECO ROMANIA SA

• **Recomandat pentru doua ore de predare.**

1. Principiul stivei: Revenirea din recursivitate (1)
2. Principiul stivei: Revenirea din recursivitate (2)
3. Ordinea de execuție a functiilor recursive
4. Formule recursive
5. Recursivitatea indirectă
6. Test grilă de evaluare a cuno°tinţelor

Recursivitatea prin exemple

Ordinea de execuție a funcțiilor recursive

Pentru următorul program, completați schema din dreapta corespunzător ordinii de apel a funcțiilor și indicați rezultatul afișat.

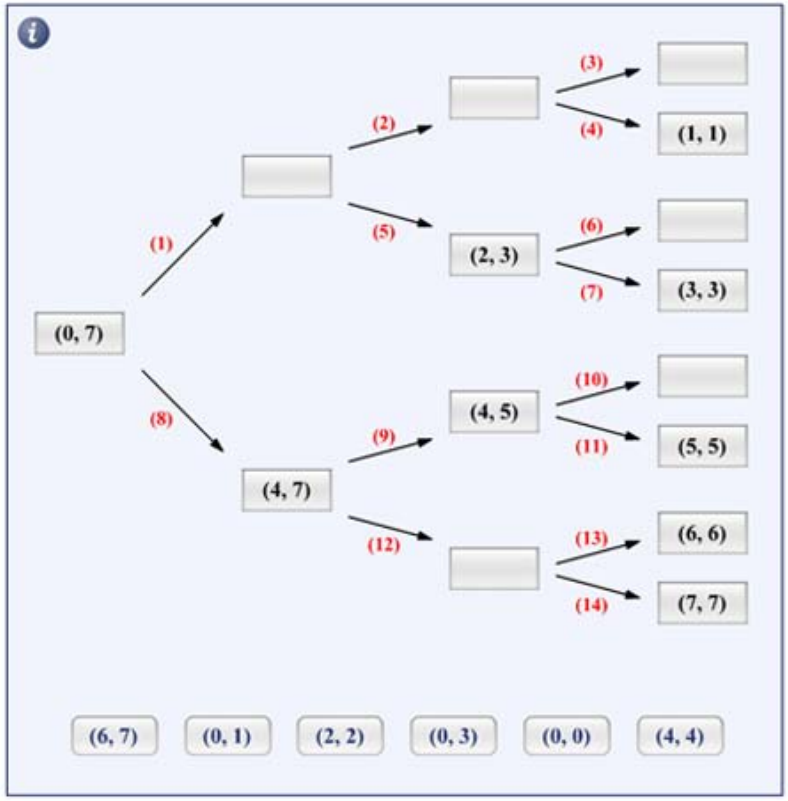
Indiciu :
Ordinea de parcurgere a schemei este indicată de cifrele de culoare roșie situate pe săgeți.

Program sursă :

```
#include <stdio.h>
int v[8] = {2, 4, 1, 3, 11, 5, 1, 3};

int impar(int n, int m)
{
    // printf("%d \t %d \n", n, m);
    if (n == m) return v[n] % 2;
    else return impar(n, (n+m)/2) +
                impar((n+m)/2 + 1, m);
}

void main()
{
    printf("%d", impar(0, 7));
}
```



informatică

SIVECO ROMANIA SA

• **Recomandat pentru patru ore de predare.**

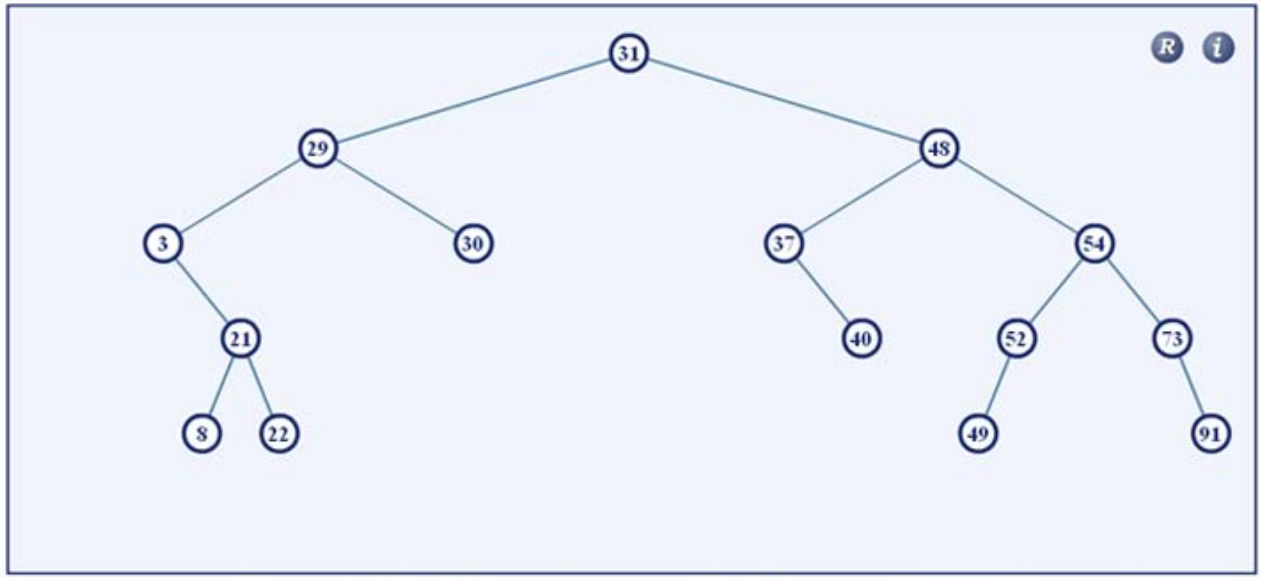
1. Arbore parțial de cost minim
2. Definirea ºi reprezentarea arborilor binari
3. Test grilă pentru evaluarea cunostintelor (1)
4. Parcurgerea arborilor binari
5. Forma poloneză asociată unei expresii aritmetice
6. Test grilă pentru evaluarea cunoºtintelor (2)
7. Arborele binar de căutare
8. Sortarea cu ansamble (HEAP SORT)
9. Selecția arboreºcentă
10. Selectia arborescenta: Turnirul
11. Test grilă pentru evaluarea cunoºtințelor (3)

Arbori binari

Arborele binar de căutare

Definiție: Se numește **arbore de căutare** un arbore binar ale cărui noduri au o cheie de identificare, iar pentru fiecare nod avem o serie de proprietăți.

Funcțiile de bază pentru un arbore binar sunt: inserarea , căutarea , ștergerea , listarea .



Funcții arbore : 1) inserare → 25 2) căutare → ? 3) ștergere 4) listare

Parcurgere :

informatică

AeL eCONTENT
Built to teach intelli gently

• Recomandat pentru zero ore de predare.

1. Noțiuni de bază: Definiții
2. Clase speciale de grafuri: Graful scară
3. Clase speciale de grafuri: Graful bipartit
4. Clase speciale de grafuri: Graful complet
5. Clase speciale de grafuri: Test grilă de evaluare a cunoºtinþelor.
6. Reprezentarea grafurilor neorientate
7. Reprezentarea grafurilor neorientate: Test grilă de evaluare a cunoºtinþelor
8. Parcurgerea grafurilor neorientate
9. Conexitate
10. Conexitate: Test grilă de evaluare a cunoºtinþelor
11. Grafuri hamiltoniene: Definiție. Jocul icosian
12. Grafuri hamiltoniene: Problema comis-voiajorului
13. Grafuri hamiltoniene: Problema cavalerilor
14. Grafuri hamiltoniene: Test grilă de evaluare a cunoºtinþelor
15. Grafuri euleriene : Definiție
16. Grafuri euleriene : Aplicație
17. Grafuri euleriene : Test grilă de evaluare a cunoºtinþelor
18. Aplicații: Colorarea muchiilor unui graf
19. Aplicații: Graf bipartit complet
20. Aplicații: Grafuri izomorfe

Grafuri neorientate

Clase speciale de grafuri : Graful complet

Definiție: Se numește **graf complet** cu n vârfuri un graf care are proprietatea că oricare două noduri diferite sunt adiacente.

Observație: Un astfel de graf, notat K_n, are $\frac{n(n-1)}{2}$ muchii.

Aplicație:

La un turneu de șah participă n șahiști. Știind că fiecare șahist trebuie să susțină câte o partidă cu fiecare din ceilalți n-1 șahiști și nu poate susține mai mult de o partidă pe zi, să se afle numărul minim de zile în care se poate termina turneul.

Indicații:

Șahiștii se pot reprezenta prin vârfurile grafului complet Kn, iar partidele desfășurate într-o zi prin muchii care nu au două câte două extremități comune.

SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelligently

• **Recomandat pentru doua ore de predare.**

1. Drumul lui Napoleon
2. Istoricul grafurilor
3. Noţiuni elementare
4. Graf parţial
5. Subgraf
6. Graf bipartit
7. Universul
8. Graf conex

Grafuri Neorientate

Lecţii pentru AEL | Built to teach intelligently

Graf bipartit

Patru persoane doresc să-şi cumpere câte un autoturism având la alegere o listă ce cuprinde 4 tipuri de maşini. Cunoscând preferinţele fiecărei persoane , să se determine graficul realizat de firma de vânzari în vederea repartizării autoturismelor prezente în stoc.

Resetare | Verificare

Maşina / Persoana	4 X 4	Decapotabilă	Cabrio	Clasică
Andrei	X		X	
Mihai			X	
Ana		X		X
Raluca	X			X

informatică | Teorie | Obiective

SIVECO ROMANIA SA

• **Recomandat pentru sase ore de predare.**

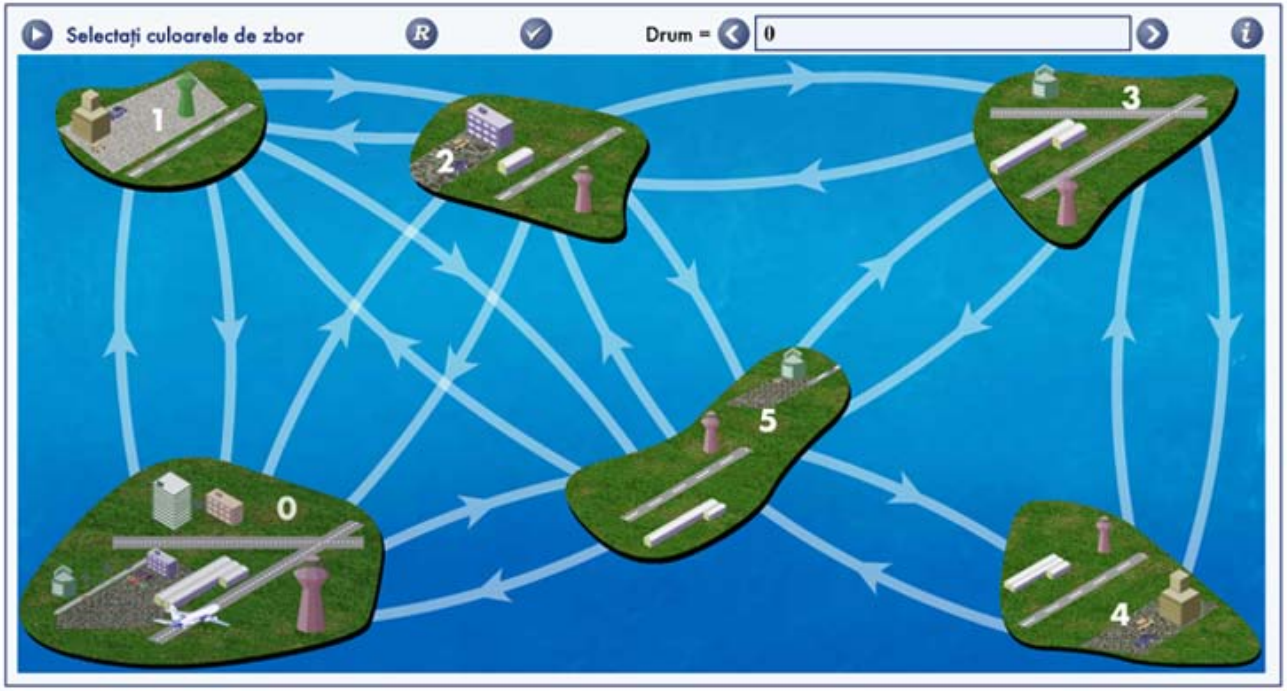
1. Noþiuni introductive
2. Lanþ, Drum, Circuit
3. Graf parþial. Subgraf
4. Graf orientat complet
5. Graful turneu
6. Test grilă de evaluare a cunoºtinþelor (1)
7. Matrice asociate grafurilor orientate
8. Algoritmul Roy-Warshall
9. Problema celebrităþii
10. Test grilă de evaluare a cunoºtinþelor (2)
11. Graf conex. Graf tare conex. Componente conexe
12. Test grilă de evaluare a cunoºtinþelor (3)
13. Graf hamiltonian
14. Costuri minime în grafuri: Algoritmul lui Dijkstra
15. Algoritmul Roy-Floyd

Grafuri orientate

Graf hamiltonian

Fie un graf $G = (X, U)$. Se numeşte **ciclu hamiltonian** un ciclu elementar care conţine toate vârfurile grafului. Se numeşte **graf hamiltonian** un graf care conţine un ciclu hamiltonian.

Pornind de pe prima insulă, vizitaţi în călătoria cu avionul toate celelalte insulele din arhipelag, trecând prin fiecare o singură dată.



SIVECO ROMANIA SA

• **Recomandat pentru cinci ore de predare.**

1. Noțiuni fundamentale. Recapitulare
2. MaxHeap. Prezentare
3. MinHeap. Prezentare
4. Test Recapitulativ
5. Inserarea unui nod într-un heap
6. Crearea unui heap prin inserări repetate
7. Combinarea a doua heap-uri
8. Crearea unui heap prin combinare
9. Extragerea nodului de valoare maximă
10. Coadă cu prioritate
11. Test Recapitulativ. Coadă cu prioritate
12. Heapsort
13. Test Recapitulativ. HeapSort

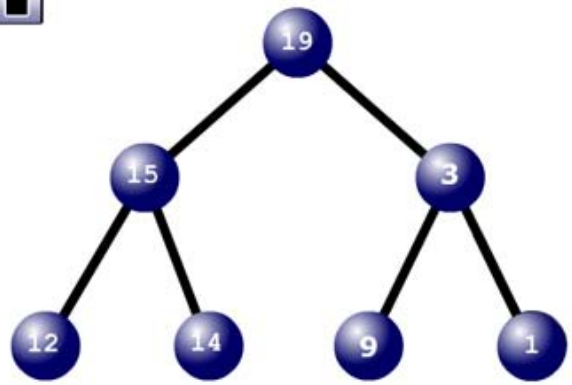
Heap

Lecții pentru AEL | Built to teach intelligently

Combinarea a două heap-uri

O astfel de operație creează la fiecare pas i heap-ul cu rădăcina H[i], combinând două heap-uri de dimensiuni apropiate, heap-ul cu radacina 2*i cu heap-ul cu rădăcina 2*i+1 și cu elementul H[i]. Nodul tată (inițial i) este "retrogradat" fiind înlocuit cu fiul cu valoarea cea mai mare până când proprietatea de heap va fi restabilită.

Control animație



Nodul i are o valoare mai mică decât a fiului său, ceea ce nu respectă proprietatea de heap. Astfel, nodul i este interschimbat cu fiul acestuia drept

```
CombHeap (int i, int n)
{
 int fiu, tata, aux;
 tata <- i; fiu <- 2*i;
 cat timp fiu<=n
   {
   daca fiu+1<=n atunci
     daca H[fiu] < H[fiu+1] fiu <-fiu+1;
   daca H[tata] < H[fiu] atunci
     {
      aux <- H[tata];
      H[tata] <- H[fiu];
      H[fiu] <- aux;
      tata <- fiu; fiu <- fiu*2;
     }
     altfel fiu <- n+1;
   }
}
```

pseudocod | C++ | pascal

Complexitate

informatică | Obiective

• **Recomandat pentru doua ore de predare.**

1. Noţiuni introductive
2. Noţiuni introductive de matematică
3. Exemple de algoritmi recursivi
4. Executarea programelor recursive
5. Aplicaţie: Mecanismul recursivităţii
6. Primele n numere naturale
7. Rezolvarea ecuaţiei de gradul 3
8. Turnurile din Hanoi

Recursivitate

Lecţii pentru AEL Built to teach intelligently

Noţiuni introductive de matematică

Definirea elementelor

descriptiv

Elementele care apar in şir au o anumită caracteristică.

Exemple:

1; 2; 2; 3; 3; 3; 4; 4; 4; 4;...

în acest şir apar pe rând toate numerele naturale, fiecare din ele repetându-se de atatea ori cât este valoarea lor

formulă generală

Elementele sunt definite printr-o formulă ce exprimă forma generală a termenilor.

Exemple:

1; 3; 5; 7;...; 2n-1;... este şirul numerelor naturale impare. Se observă o corespondentă între rangul termenului şi valoarea lui:
primul termen - rang 1 - valoare 2*1-1=1
al doilea termen - rang 2 - valoare 2*2-1=3.....

recursiv

Elementele sunt definite cu ajutorul unei relatii între termenul general şi termenii precedenţi.

Exemple:

→ factorialul unui număr: $n! = \begin{cases} 1, n = 0 \\ n \times (n-1)!, n \geq 1 \end{cases}$

→ şirul lui Fibonacci:

$$fib(n) = \begin{cases} 1, n = 0 \\ fib(n-1) + fib(n-2), n > 1 \end{cases}$$

Noţiunea de şir

Şirul este o succesiune de valori numite **elementele şirului**, aşezate ordonat după o ordine bine definită.

În şirul a_p, a_{p+1}, a_{p+2},...,a_n,... fiecare element are o pozitie fixată numită **rangul elementului.**

informatică | Obiective

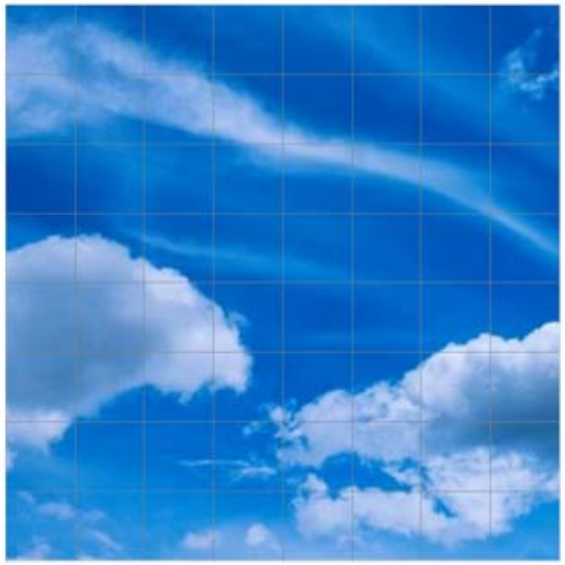
- **Recomandat pentru trei ore de predare.**

1. Algoritmul lui Dijsktra: Prezentarea problemei
2. Algoritmul lui Dijsktra: Reprezentarea informațiilor
3. Algoritmul lui Dijsktra: Implementare
4. Algoritmul lui Dijsktra: Evaluare

Drumuri minime în graf

AeL eCONTENT
Built to teach *intelli* gently

Algoritmul lui Dijkstra – reprezentarea informațiilor



Putem asocia problemei un graf orientat astfel:

- vârfurile grafului sunt aeroporturile; acestea vor fi numerotate de la 1 la N (unde N reprezintă numărul de aeroporturi);
- există arc de la vârful x la vârful y, dacă de la aeroportul corespunzător vârfului x există un zbor (arc) având ca destinație aeroportul corespunzător vârfului y.

Graful orientat asociat problemei este ponderat, deoarece putem asocia o funcție de cost, definită pe mulțimea arcelor cu valori în mulțimea numerelor reale strict pozitive, costul unui arc fiind egal cu costul zborului modelat de arcul respectiv.

Aeroportul de start este un vârf special al grafului, denumit vârf de start.

În limbaj de grafuri, problema poate fi formulată astfel:

Fie $G=(X, V)$ un graf orientat, $c: V \to R_+$ o funcție de cost, prin care se asociază fiecărui arc o valoare reală strict pozitivă și x_0 un vârf din graf. Să se determine câte un drum de cost minim de la x_0 la fiecare dintre celelalte vârfuri ale grafului.

Costul unui drum este egal cu suma costurilor arcelor din care este constituit drumul.

Unelte | Matricea costurilor | Graful asociat | Reprezentarea informațiilor

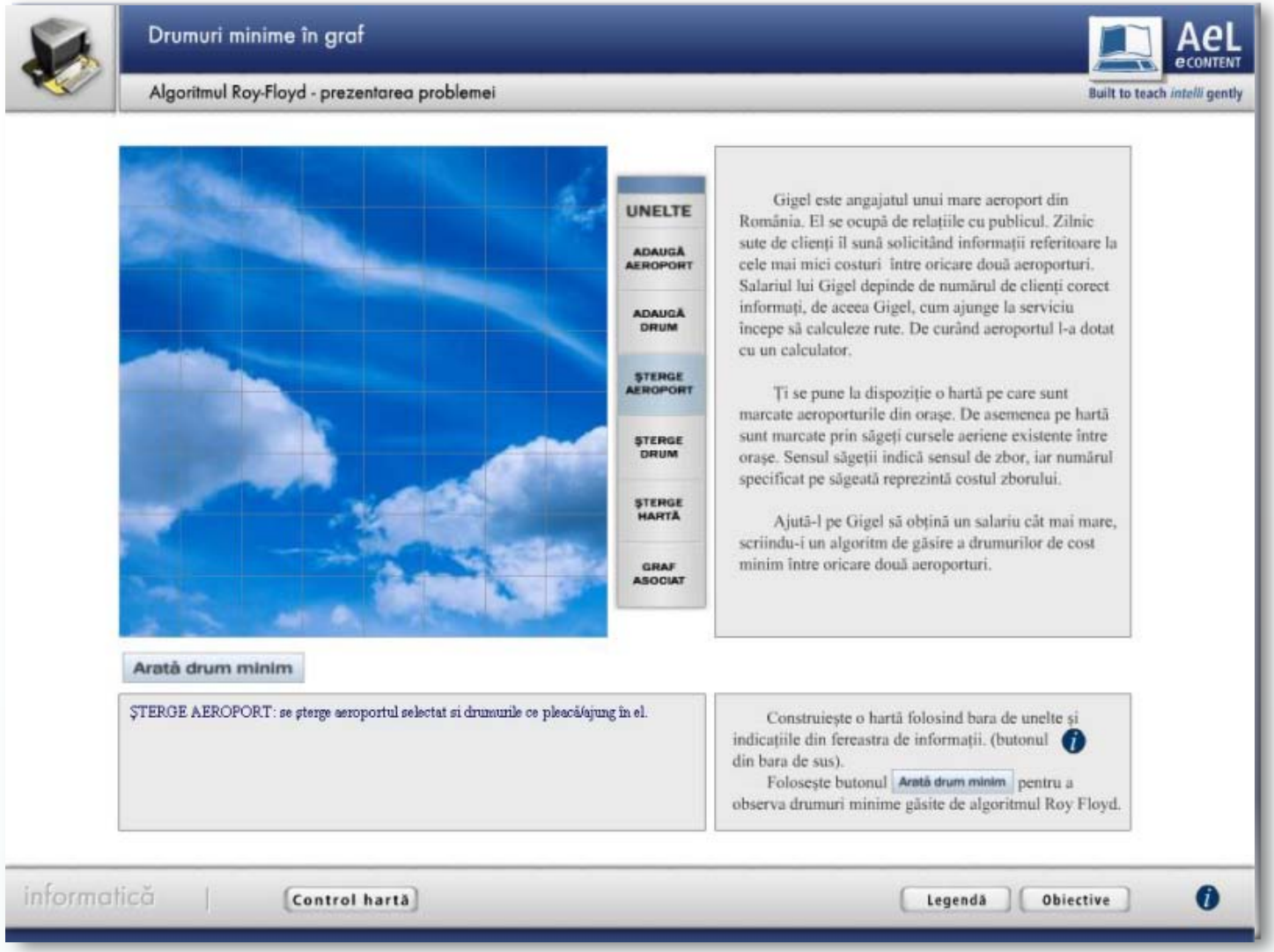
informatică | Control hartă | Definiții | Legendă | Obiective

Drumuri minime în graf. Partea II

AeL eCONTENT
Built to teach *intelli* gently

- **Recomandat pentru doua ore de predare.**

1. Algoritmul lui Roy-Floyd: Prezentarea problemei
2. Algoritmul lui Roy-Floyd: Reprezentarea informatiilor
3. Algoritmul lui Roy-Floyd: Implementare
4. Algoritmul lui Roy-Floyd: Evaluare



• **Recomandat pentru o ora de predare.**

1. Prezentare generală: Firma domnului Ionescu
2. Situaþie pratică: Aplicaþia 1
3. Captarea atentiei

• **Recomandat pentru o ora de predare.**

1. Prezentarea problemei în limbaj de grafuri
2. Test de evaluare

APM • Arbori partiali de cost minim

Prezentarea problemei în limbaj de grafuri

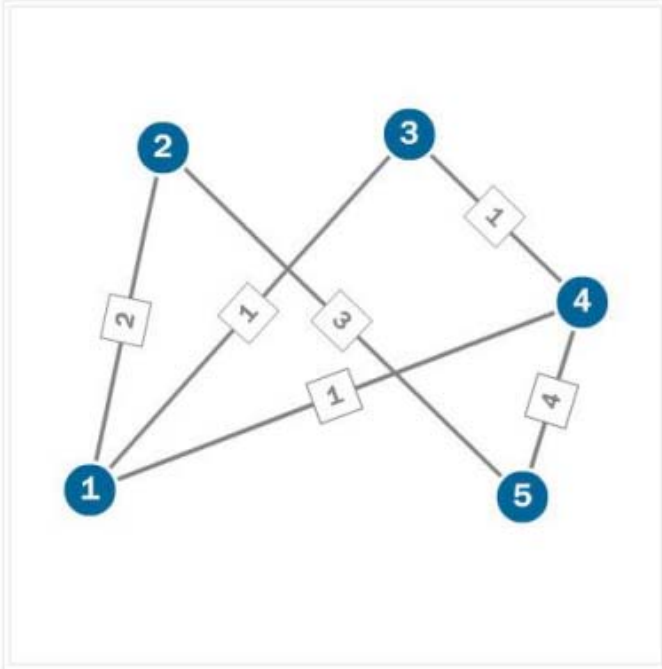
Enunţ: Fie G = (X, U) un **graf neorientat conex** şi c: U → R+ o funcţie care asociază fiecărei muchii din graf un număr real pozitiv denumit costul muchiei.

Definim costul unui **arbore parţial** ca fiind suma costurilor muchiilor arborelui.

Cerinţă: Să se determine un arbore parţial de cost minim al grafului G.

Exemple

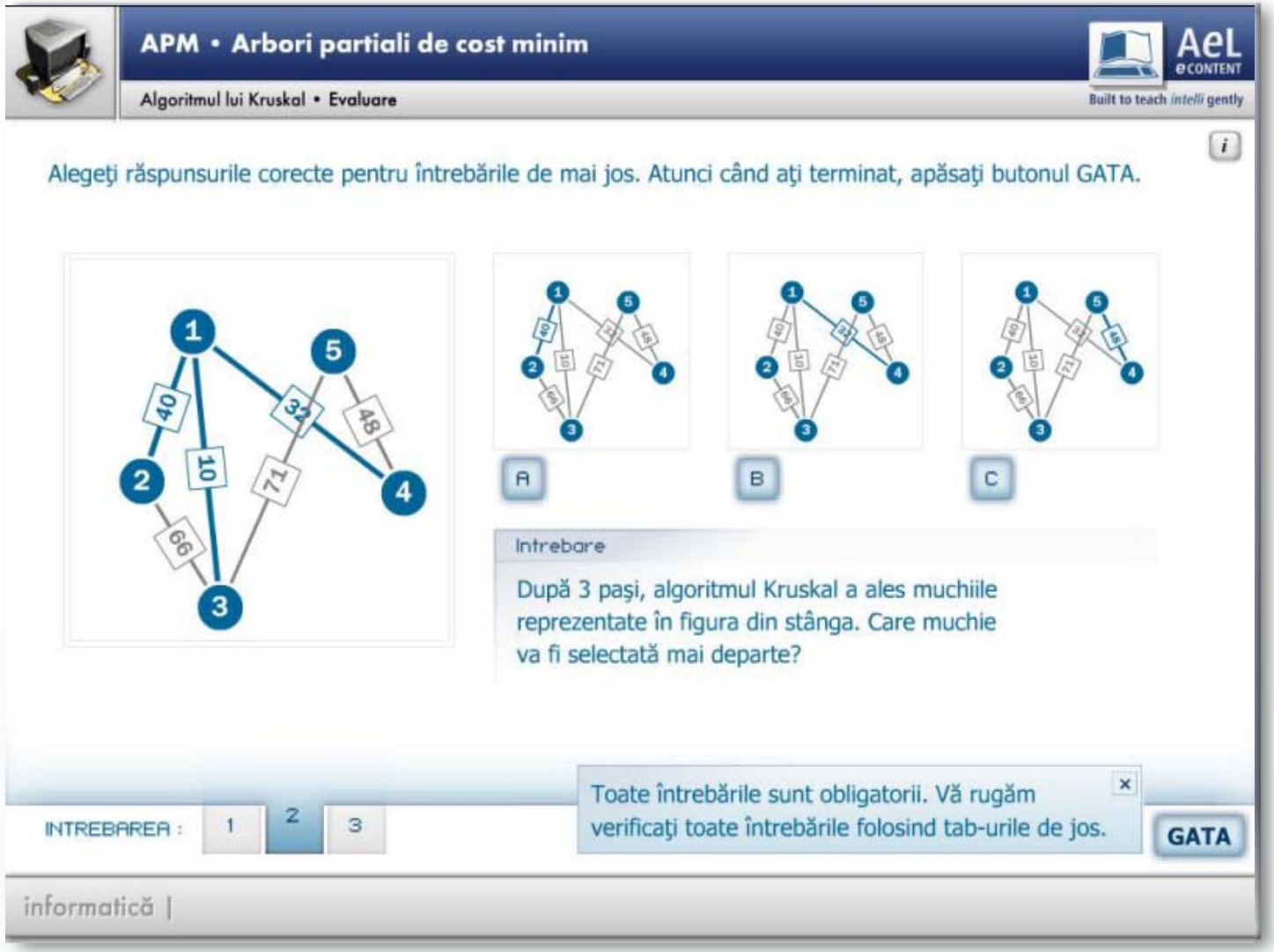
Graful din imagine admite mai mulţi arbori parţiali de cost minim: **exemplul1** **exemplul2**



informatică |

• Recomandat pentru o ora de predare.

1. Algoritmul lui Kruskal: Prezentare
2. Algoritmul lui Kruskal: Aplicaţie
3. Algoritmul lui Kruskal: Evaluare

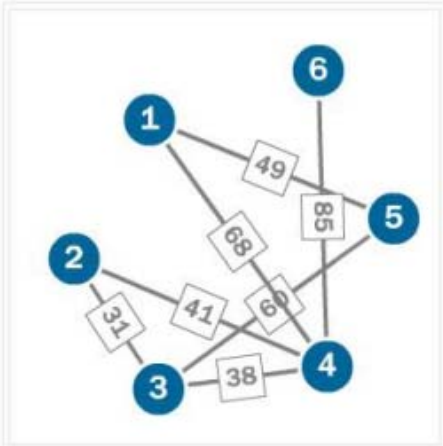


• **Recomandat pentru o ora de predare.**

1. Algoritmul lui Kruskal: Implementare
2. Algoritmul lui Kruskal: Pseudocod
3. Algoritmul lui Kruskal: Evaluare

APM • Arbori partiali de cost minim

Algoritmul lui Kruskal • Pas 1. Iniţializare



Mod de folosire a editorului

Adăugare nod: Triplu-click pe spaţiu alb
Mutare nod: Trageţi (drag) de nod
Ştergere nod: Triplu-click pe un nod
Editare număr nod: Click pe nod

Adăugare / ştergere muchie:
Faceţi dublu-click pe un nod, ţineţi apăsat şi apoi trageţi până la celălalt nod. (Maxim **9**!)
Editare cost muchie: Click pe eticheta muchiei

Puteţi edita graful din imagine. Dacă doriţi puteţi genera un graf aleatoriu, apăsând pe butonul **Generează** .

Control

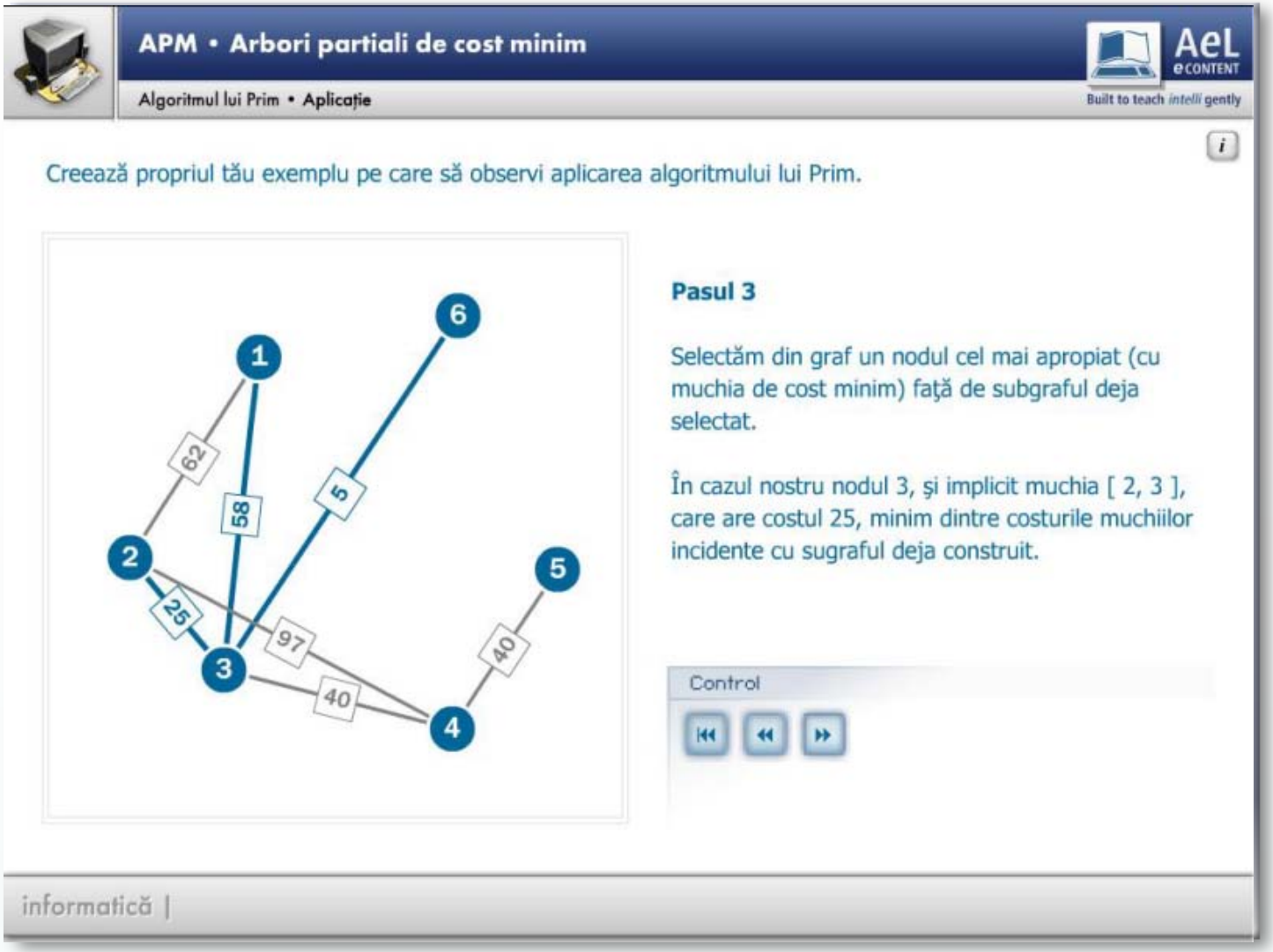
Declaraţii
C / C++ | Pascal | Pseudocod
Complexitate | Corectitudine

```
n: 6      m: 7      i: 1      NrMSel: 0
c: 1, 2, 3, 4, 5, 6
A:
G: { 1, 5, 49 }, { 5, 3, 60 }, { 3, 2, 31 },
   { 2, 4, 41 }, { 4, 6, 85 }, { 1, 4, 68 },
   { 3, 4, 38 }
```

informatică |

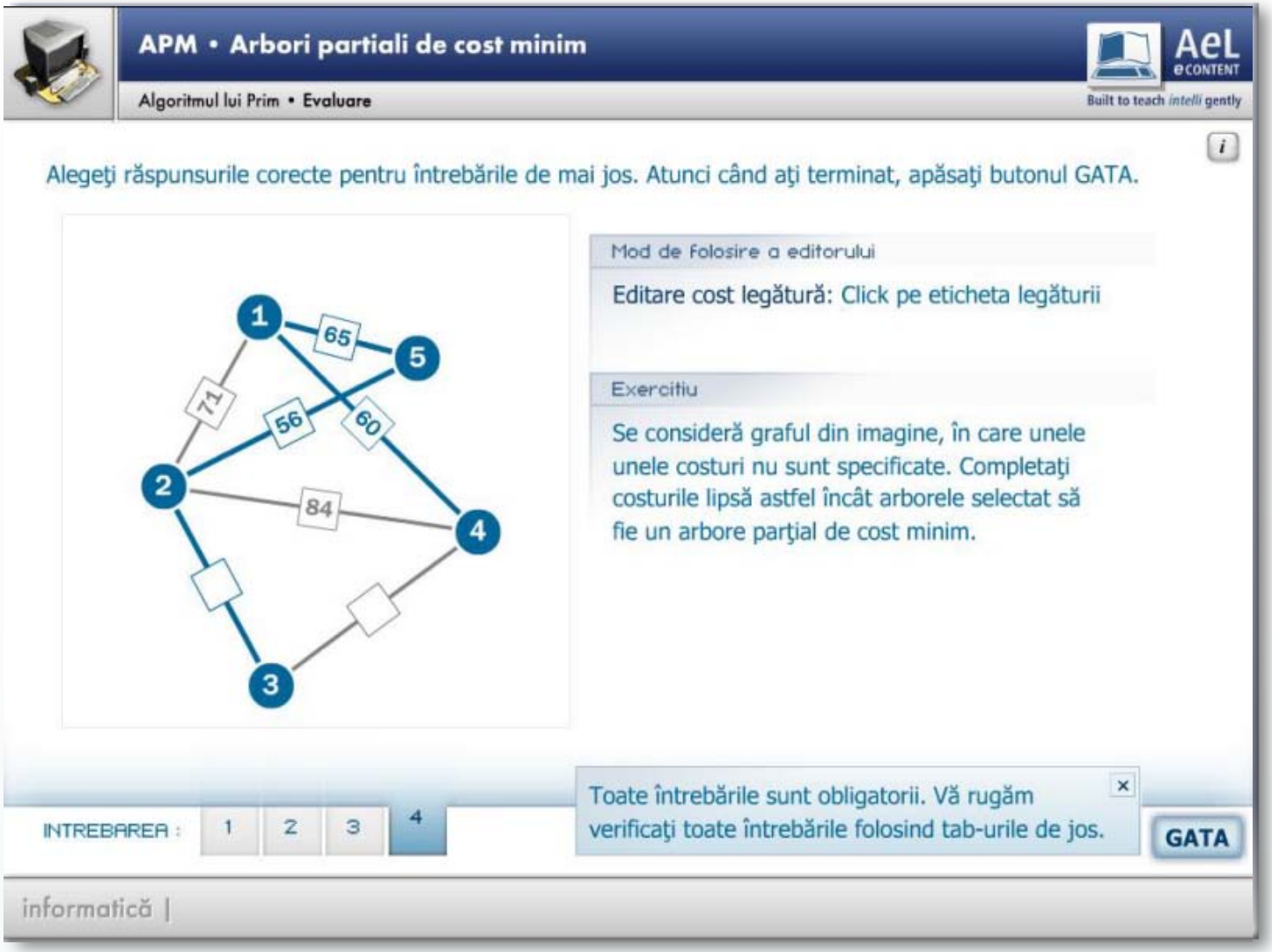
• **Recomandat pentru o ora de predare.**

1. Algoritmul lui Prim: Prezentare
2. Algoritmul lui Prim: Aplicaţie
3. Algoritmul lui Prim: Evaluare



• Recomandat pentru o ora de predare.

1. Algoritmul lui Prim: Reprezentarea informatiilor
2. Algoritmul lui Prim: Pseudocod
3. Algoritmul lui Prim: Evaluare



AeL eCONTENT
Built to teach intelli gently

• Recomandat pentru o ora de predare.

1. Teorie
2. Evaluare
3. Desenarea unui graf

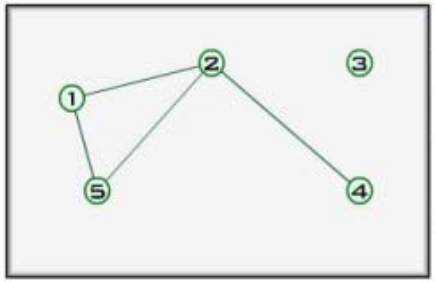
Adiacență. Incidență. Grad.

Evaluare

AeL eCONTENT
Built to teach intelli gently

EXERCITIU

➤ În graful de mai jos avem:



a) Nodurile 2 şi 5 se numesc _____

b) Nodul 4 este un nod _____

c) Pentru nodul 2 putem spune că 3 reprezintă _____ său

d) Nodul 3 este un nod _____

e) Muchia (2,5) şi nodul 2 se numesc _____

izolat | terminal | adiacente | incidente | gradul

Verifică

SIVECO ROMANIA SA

• **Recomandat pentru o ora de predare.**

1. Metode de reprezentare
2. Aplicaþie: Matricea de adiacenþã
3. Aplicaþie: Lista vecinilor
4. Secvenþe de cod

Reprezentare grafuri neorientate

Metode de reprezentare

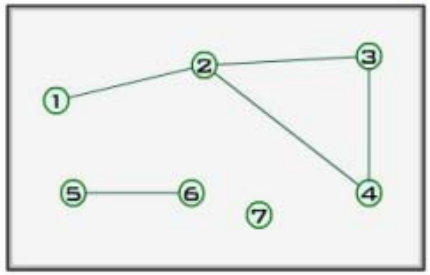
DEFINIŢII

➤ **Matricea de adiacenţă** este o matrice **a** cu **n** linii şi **n** coloane, în care elementele **a[i,j]** se definesc astfel:

$$a[i,j] = \begin{cases} 1, \text{ dacă } \exists \text{ muchia } [i,j] \text{ cu } i \neq j \\ 0, \text{ în caz contrar} \end{cases}$$

➤ **Lista vecinilor nodului x** cuprinde toate nodurile care sunt extremităţi ale muchiilor ce trec prin nodul **x**.

EXEMPLU



Matricea de adiacenţă

	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	1	0	1	0	0	0
4	0	1	1	0	0	0	0
5	0	0	0	0	0	1	0
6	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0

Lista vecinilor

nodul	*lista vecinilor*
1	2
2	1,3,4
3	2,4
4	2,3
5	6
6	5
7	

Vectorul de muchii U={(1,2), (2,3), (2,4), (3,4), (5,6)}

1/2

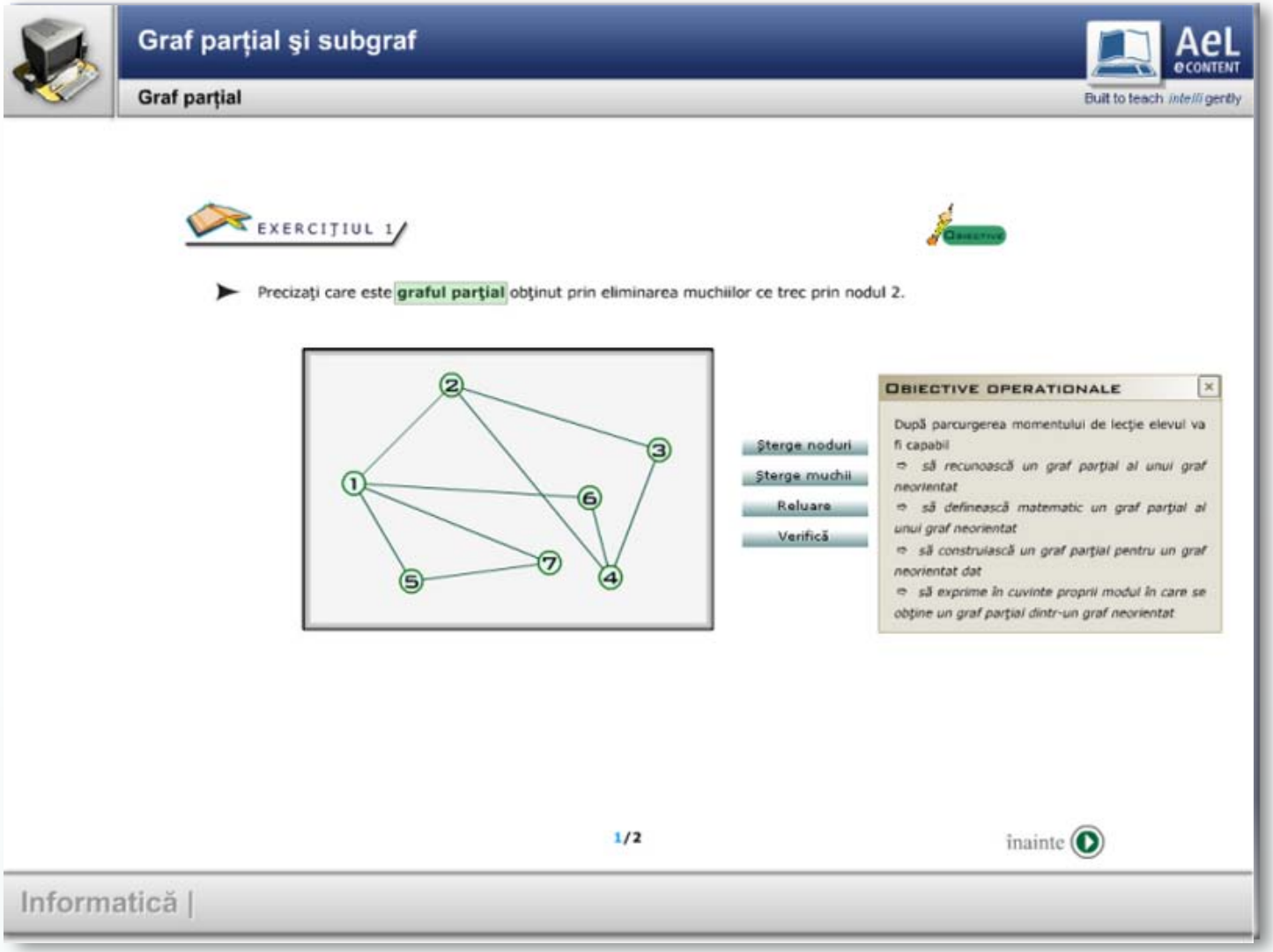
înainte

Informatică |

Graf partial °i subgraf (1)

- **Recomandat pentru o ora de predare.**

1. Graf parțial
2. Subgraf



AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

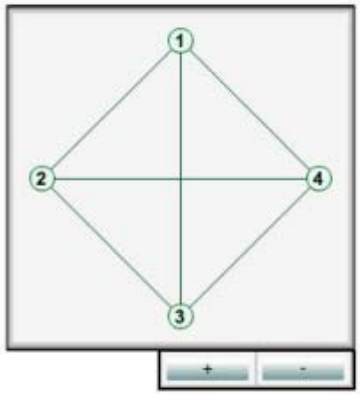
1. Graf complet
2. Graf bipartit
3. Graf planar

Tipuri de grafuri

Graf complet

EXEMPLU (Graf complet)

Graful K_4



OBSERVAȚIE

➤ Un graf complet cu **n** vârfuri, are $\frac{n \cdot (n-1)}{2}$ muchii.

1/2

înainte

SIVECO ROMANIA SA

• **Recomandat pentru o ora de predare.**

1. Lanþ
2. Ciclu
3. Arbori
4. Evaluare
5. Secvenþe de cod

Lanț. Ciclu. Arbori

Lanț

AeL eCONTENT
Built to teach *intelli* gently

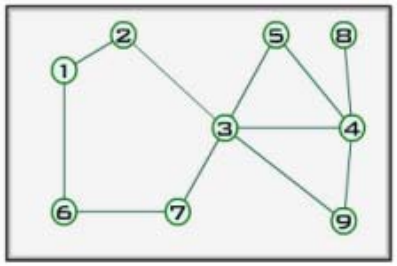
DEFINIŢIE

Se numeşte **lanţ** în graful G, o succesiune de vârfuri $L=(z_1,z_2,\ldots,z_k)$, unde $z_1,z_2,\ldots,z_k \in X$, cu proprietatea că oricare două vârfuri consecutive sunt adiacente, adică există muchiile $[z_1,z_2], [z_2,z_3],\ldots,[z_{k-1},z_k] \in U$.

EXEMPLU

I

În graful alăturat putem distinge lanţurile:



$L_1=\{1,2,3,5,4,8\}$ care este **elementar**, **simplu** şi de **lungime** 5

$L_2=\{1,2,3,2\}$ care este **neelementar** şi de **lungime** 3

$L_3=\{9,3,5,4,3,2\}$ care este **neelementar**, **simplu** şi de **lungime** 5

$L_4=\{6,7,3,5,4,8\}$ care este **elementar**, **simplu** şi de **lungime** 5

1/3

înainte

Informatică |

• **Recomandat pentru o ora de predare.**

1. Noþiuni referitoare la graful orientat
2. Construirea unui graf orientat
3. Evaluare

Adiacență. Incidență. Grad.

Evaluare

EXERCITIU

Completaţi frazele de mai jos cu unul din cuvintele disponibile.

a) Dacă (x,y) este arc, atunci x este _____ al lui y.

b) Arcul (x,y) şi nodul x se numesc _____

c) Un arc de forma _____ se numeşte _____

d) Gradul _____ al nodului x se notează cu _____

e) Mulţimea de _____ ai lui x se notează cu _____

interior | $\Gamma^{+}(x)$ | *succesori* | *bucla* | *(x, x)* | $d^{-}(x)$ | *predecesor* | *incidente*

Verifică

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Graf parțial
2. Subgraf

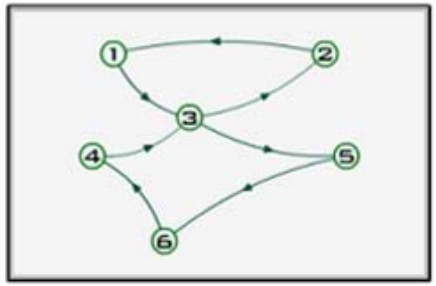
Graf parțial și subgraf

Subgraf

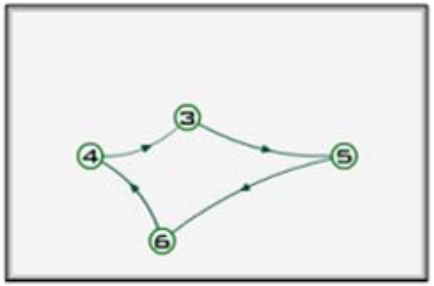
AeL eCONTENT
Built to teach intelli gently

EXEMPLU

Se consideră graful G de mai jos în care X={1, 2, 3, 4, 5, 6} și U={(2,1), (1,3), (3,2), (4,3), (3,5), (6,4), (5,6)}. Construim **subgraful** G_1=(Y,V), unde Y={3, 4, 5, 6} și V={(4,3), (3,5), (6,4), (5,6)} adică din graful G au fost eliminate nodurile 1 și 2 și arcele incidente acestora.



Graful G=(X,U)



Graful G_1=(Y,V)

1/2

înainte

SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Teorie
2. Exemplu
3. Exercițiu
4. Evaluare

Matricea de adiacență

Teorie

AeL eCONTENT
Built to teach intelli gently

DEFINIŢIE

➤ **Matricea de adiacenţă** este o matrice **a** cu **n** linii şi **n** coloane, în care elementele **a[i,j]** se definesc astfel:

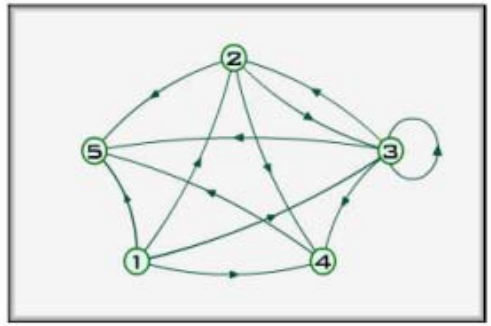
$$a[i,j] = \begin{cases} 1, \text{ dacă } \exists \text{ arcul } (i,j) \text{ în mulţimea } U \\ 0, \text{ în caz contrar} \end{cases}$$

EXEMPLU

➤ Deplasaţi mouse-ul deasupra elementelor grafului alăturat şi urmăriţi matricea de adiacenţă.

Matricea de adiacenţă

	1	2	3	4	5
1	0	1	1	1	1
2	0	0	1	1	1
3	0	1	1	1	1
4	0	0	0	0	1
5	0	0	0	0	0



SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Definire
2. Exerciţii

Matricea vârfuri - arce

Definire

DEFINIŢIE

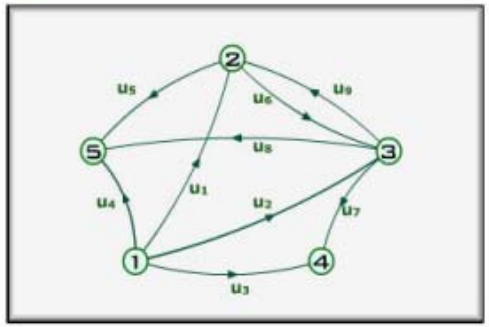
➤ Matricea **vârfuri-arce** este o matrice **B** cu **n**=|X| linii şi **m**=|U| coloane, în care fiecare element b[i,j] este:

- 1, dacă nodul i este o extremitate iniţială a arcului u_j;
- -1, dacă nodul i este o extremitate finală a arcului u_j;
- 0, dacă nodul i nu este o extremitate a arcului u_j.

EXEMPLU

Matricea vârfuri-arce

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & -1 & 1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \end{pmatrix}$$



1/2

înainte

Informatică |

SIVECO ROMANIA SA

• **Recomandat pentru o ora de predare.**

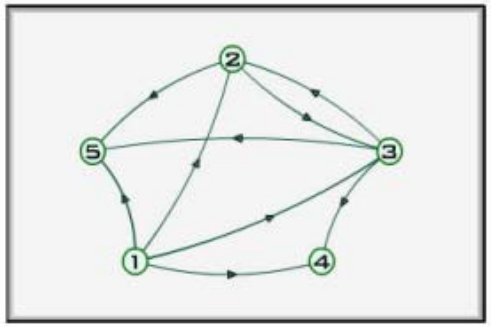
1. Definire
2. Exerciţii

Listele vecinilor

Definire

EXEMPLU

➤ Deplasaţi mouse-ul deasupra nodurilor grafului şi analizaţi **lista succesorilor** şi **lista predecesorilor** fiecărui nod.



Listele vecinilor

nodul	$\Gamma^+(x)$	$\Gamma^-(x)$
1	2,3,4,5	
2	3,5	1,3
3	2,4,5	1,2
4		1,3
5		1,2,3

OBSERVAŢIE

➤ Numărul de elemente din lista succesorilor lui x reprezintă gradul exterior al nodului x, iar numărul de elemente din lista predecesorilor lui x reprezintă gradul interior al nodului x.

1/2

înainte

Informatică |

SIVECO ROMANIA SA

• **Recomandat pentru o ora de predare.**

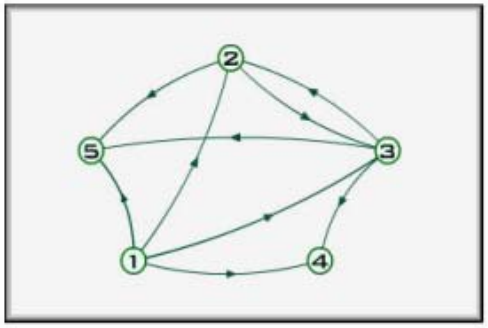
1. Definire
2. Exerciţii

Vectorul de arce

Definire

EXEMPLU

Analizaţi arcele grafului de mai jos şi vectorul de arce U.



Vectorul de arce

U={(1,2), (1,3), (1,4), (1,5), (2,3), (2,5), (3,2), (3,4), (3,5)}

1/2

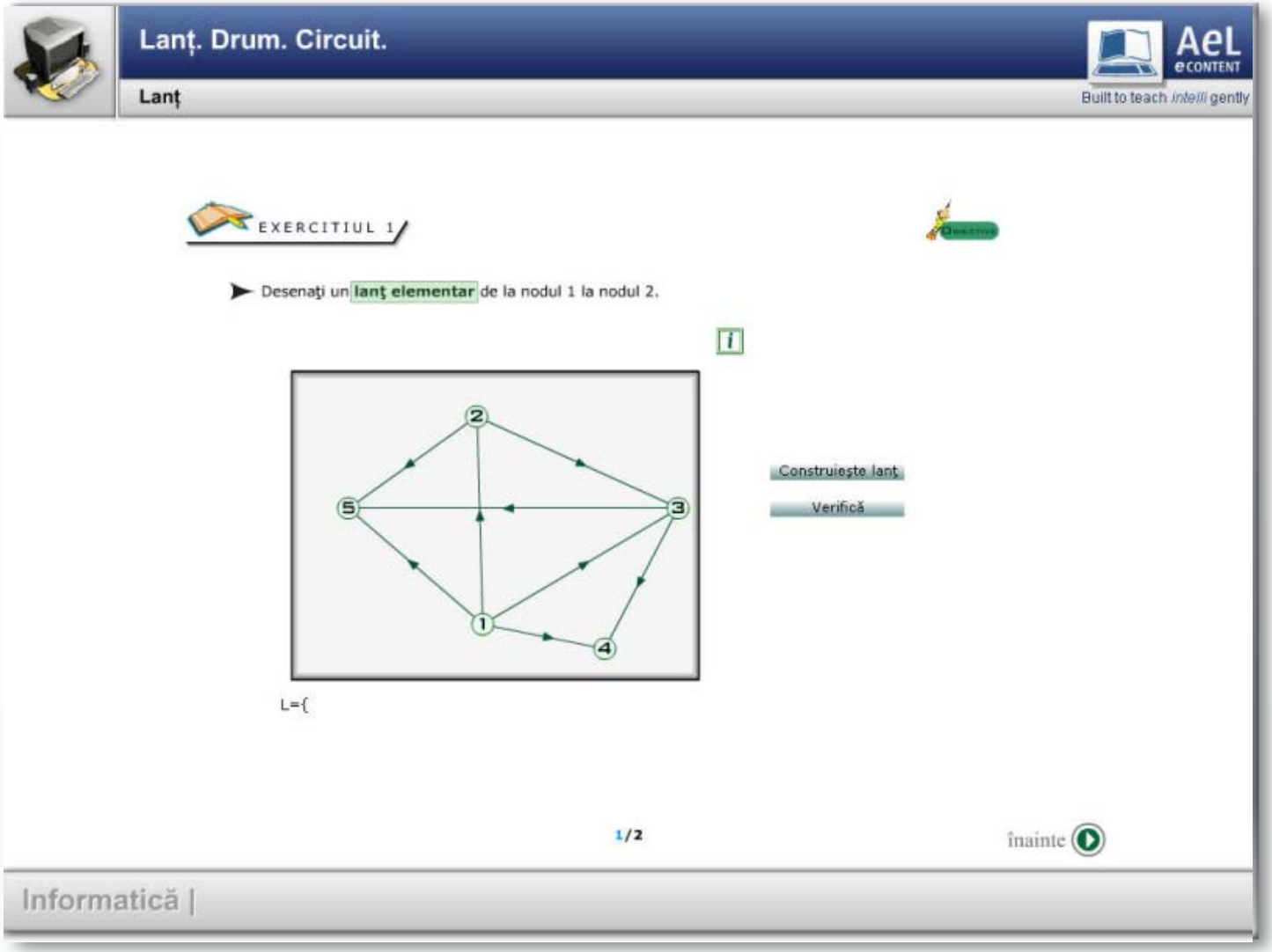
înainte

Informatică |

SIVECO ROMANIA SA

• Recomandat pentru o ora de predare.

1. Lanþ
2. Drum
3. Circuit



AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Definire
2. Exercițiu

Matricea Drumurilor

Definire

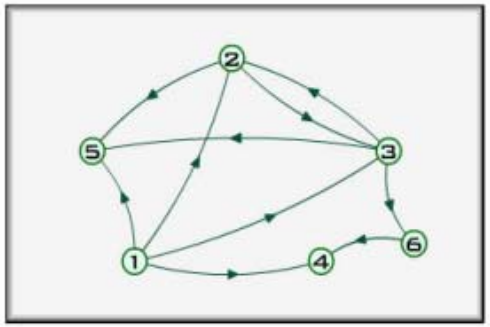
AeL eCONTENT
Built to teach intelli gently

DEFINIȚIE

➤ **Matricea drumurilor** este o matrice **D** cu **n** linii şi **n** coloane, în care: $d[i,j] = \begin{cases} 1, \text{ dacă există drum de la } i \text{ la } j \\ 0, \text{ în caz contrar} \end{cases}$

EXEMPLU

➤ Pentru graful de mai jos urmăriţi cum arată matricea de adiacenţă şi matricea drumurilor.



Matricea de adiacenţă

	1	2	3	4	5	6
1	0	1	1	1	1	0
2	0	0	1	0	1	0
3	0	1	0	1	1	1
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	1	0	0

Matricea drumurilor

	1	2	3	4	5	6
1	0	1	1	1	1	1
2	0	1	1	1	1	1
3	0	1	1	1	1	1
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	1	0	0

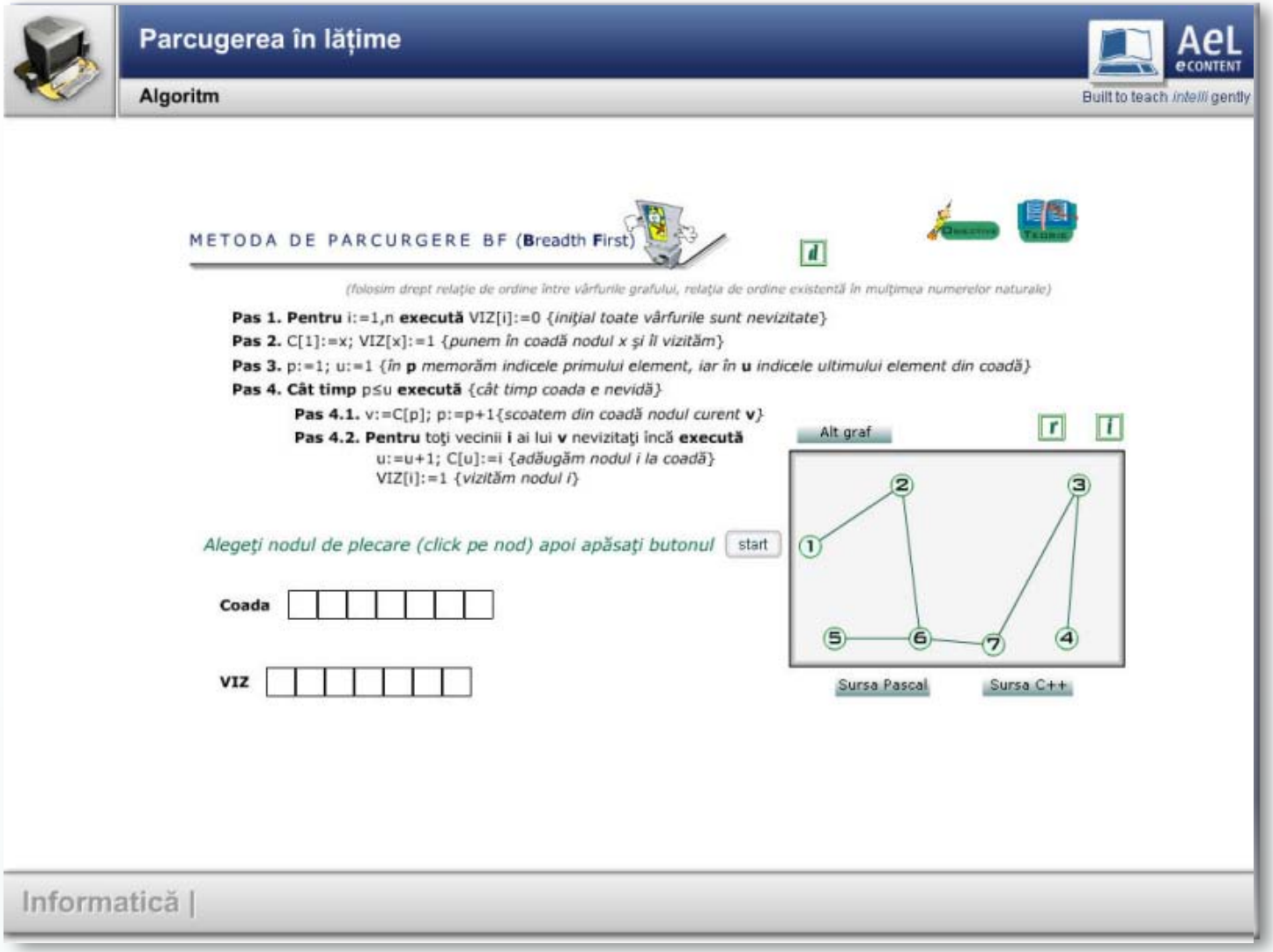
înainte

SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Algoritm
2. Exemplu
3. Exercițiu

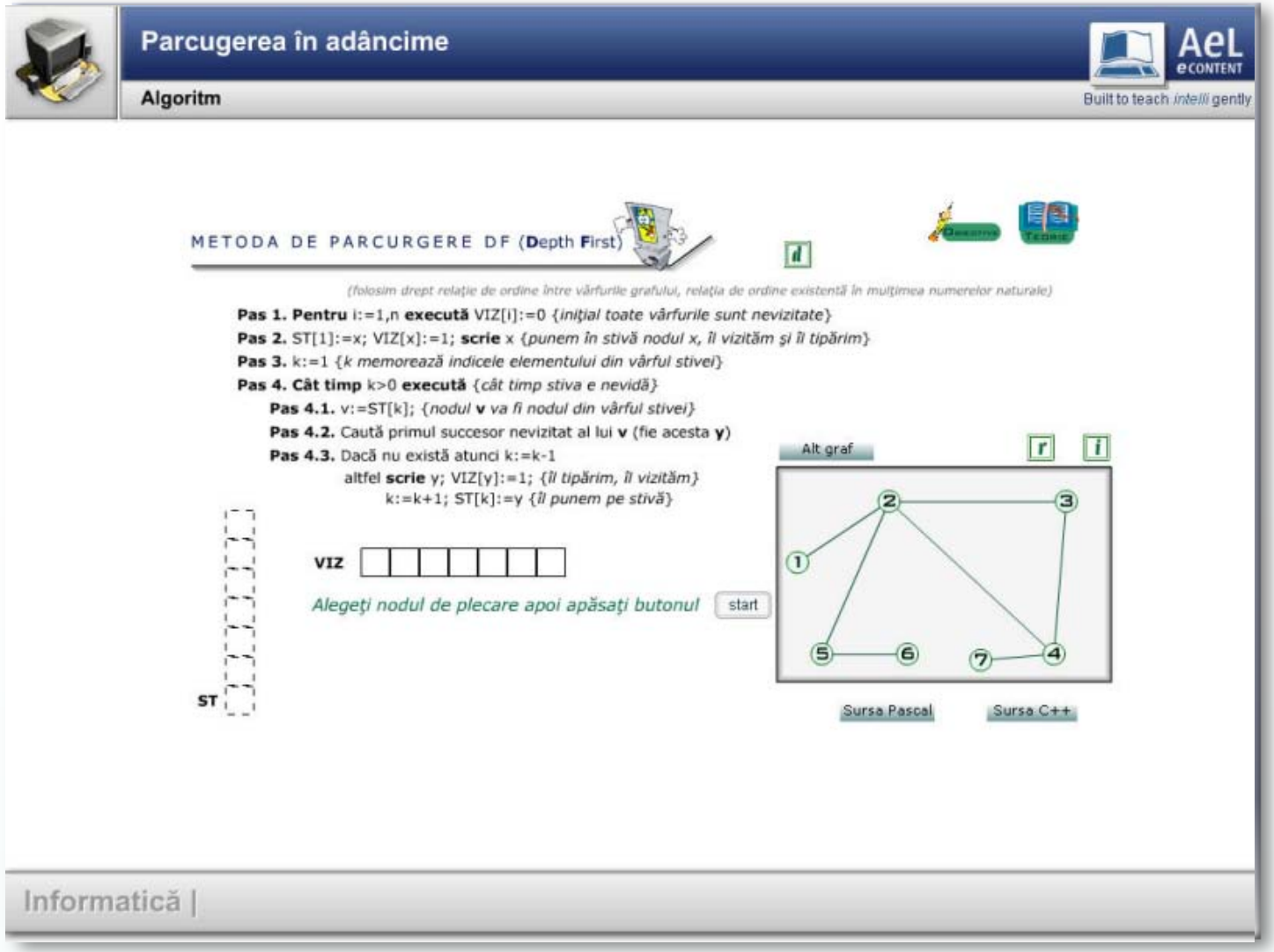


SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Algoritm
2. Exemplu
3. Exercițiu

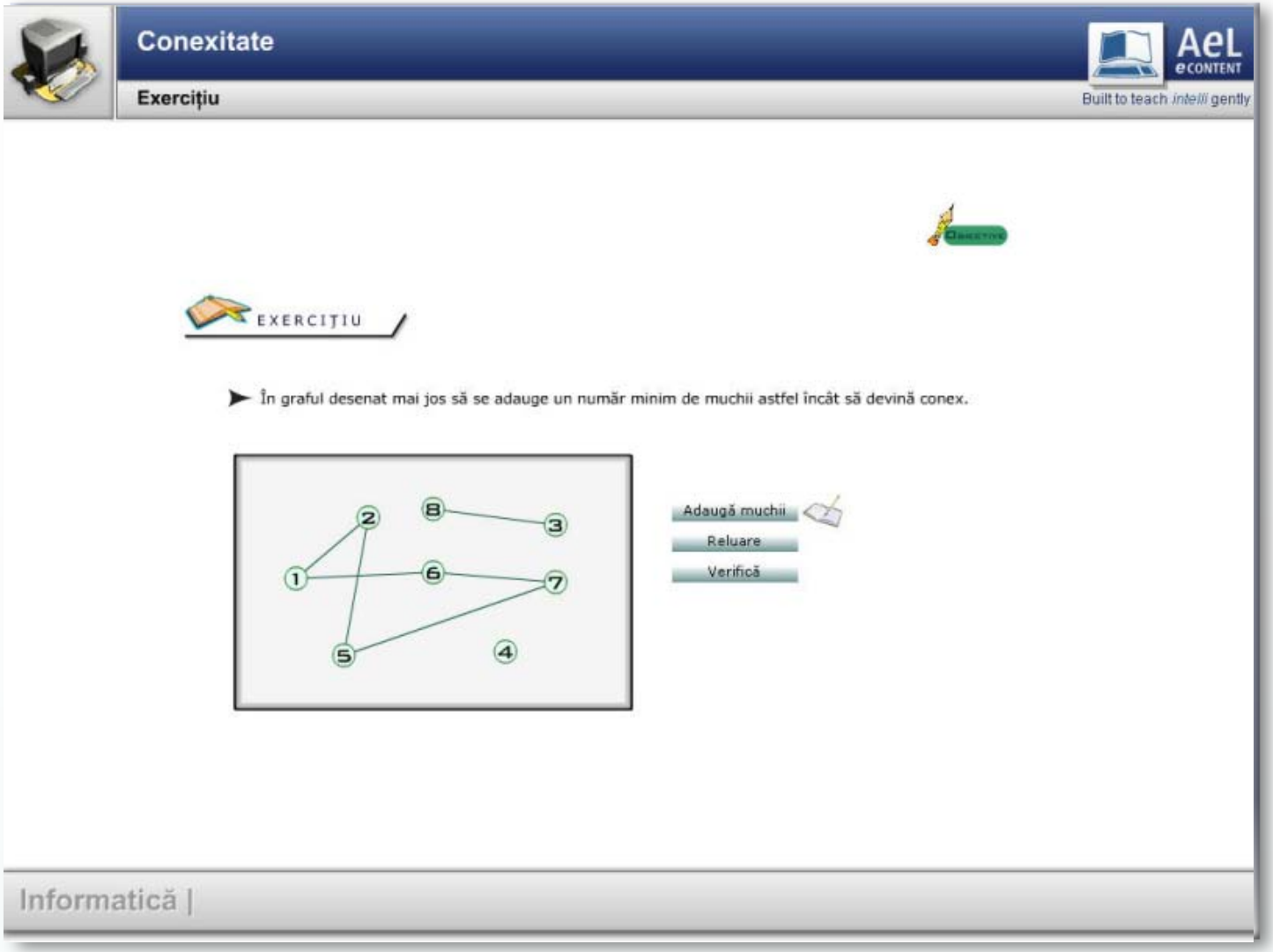


SIVECO ROMANIA SA

AeL eCONTENT
Built to teach *intelli* gently

• **Recomandat pentru o ora de predare.**

1. Definire
2. Exercițiu
3. Exercițiu
4. Secvențe de cod



SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Jocul Icosian
2. Exemple grafuri hamiltoniene si euleriene. Problema podurilor din Konisberg
3. Grafuri complete euleriene
4. Exerciții

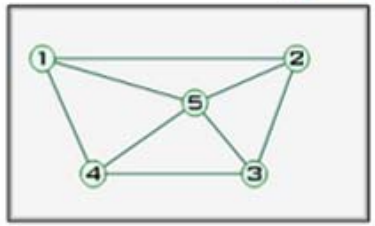
Grafuri hamiltoniene şi euleriene

AeL eCONTENT
Built to teach intelli gently

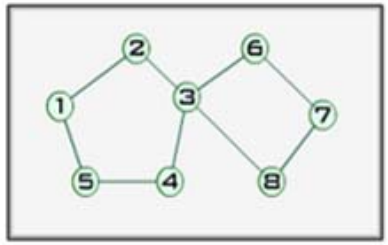
Exemple grafuri hamiltoniene şi euleriene. Problema podurilor din Konisberg

EXEMPLUL 1

Graful alăturat este **hamiltonian**, deoarece ciclul C=[1, 2, 3, 5, 4, 1] este **ciclu hamiltonian** (pleacă din vârful 1 şi se încheie tot în 1 şi trece prin fiecare vârf o singură dată, iar muchiile [1, 2], [2, 3], [3, 5], [5, 4] şi [4,1] sunt distincte două câte două).



EXEMPLUL 2

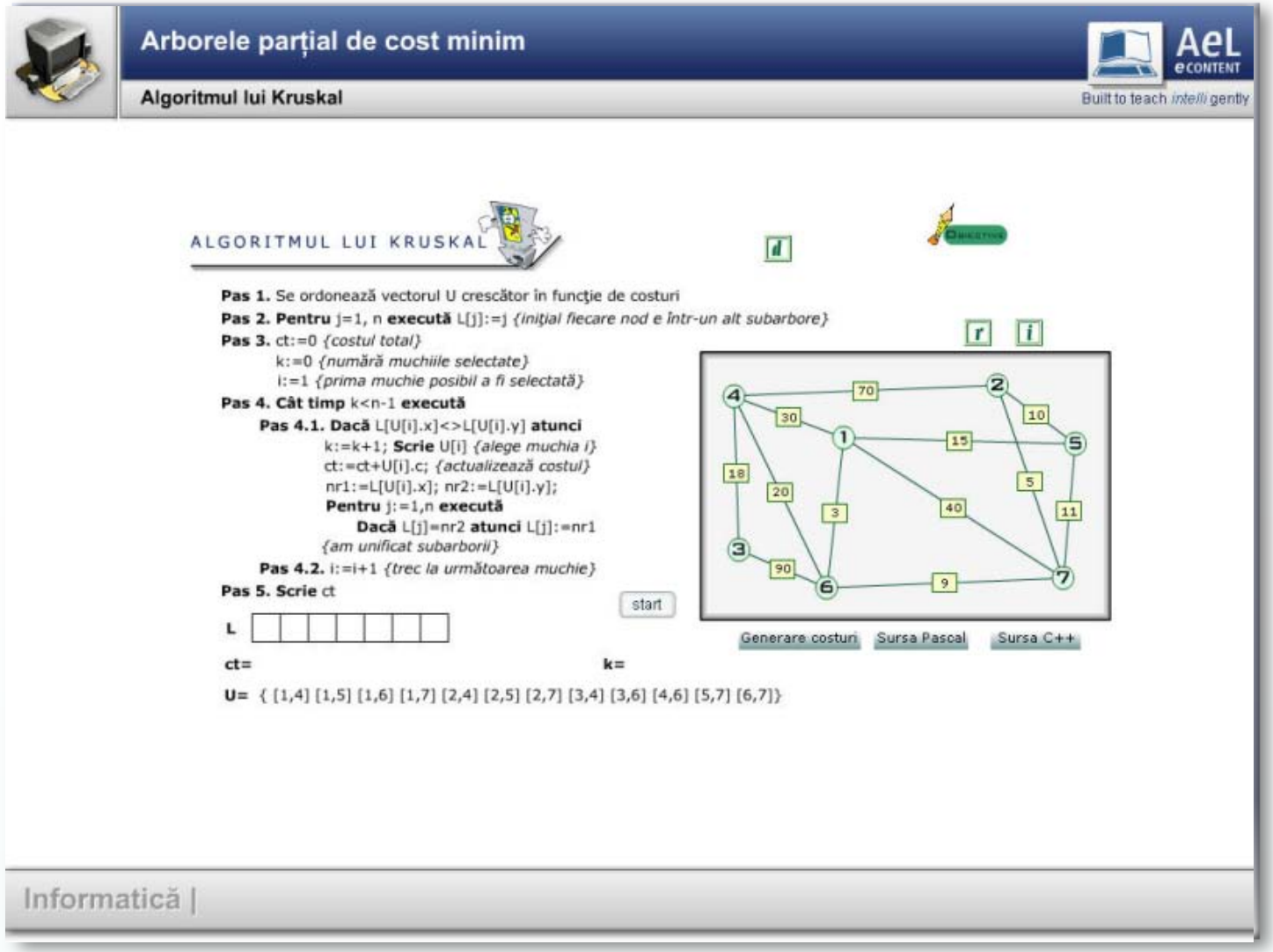


Graful alăturat este **eulerian** deoarece ciclul C=[1, 2, 3, 6, 7, 8, 3, 4, 5, 1] este **ciclu eulerian** (pleacă din vârful 1 şi se termină tot în 1 şi acest ciclu conţine toate muchiile).

SIVECO ROMANIA SA

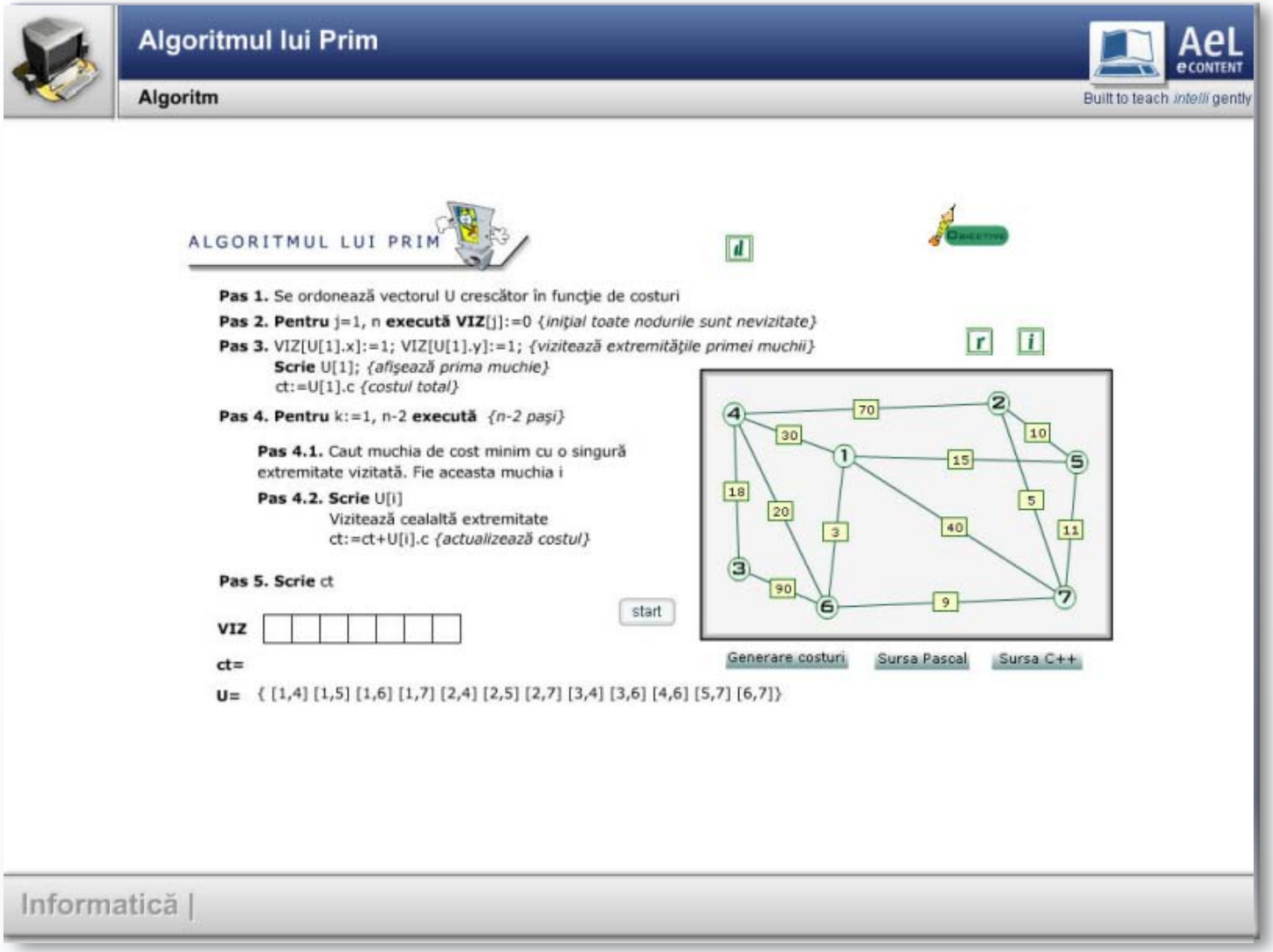
• **Recomandat pentru o ora de predare.**

1. Definire
2. Algoritmul lui Kruskal
3. Exerciţiu



• **Recomandat pentru o ora de predare.**

1. Algoritm
2. Exercițiu
3. Exercițiu



SIVECO ROMANIA SA

AeL eCONTENT
Built to teach intelli gently

• **Recomandat pentru o ora de predare.**

1. Definire
2. Exercițiu

Tare conexitate

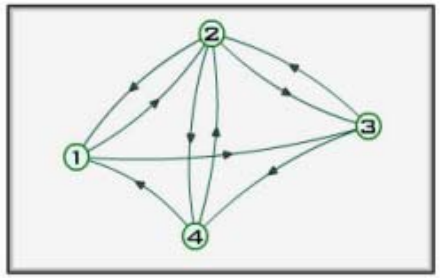
Definire

DEFINIȚIE

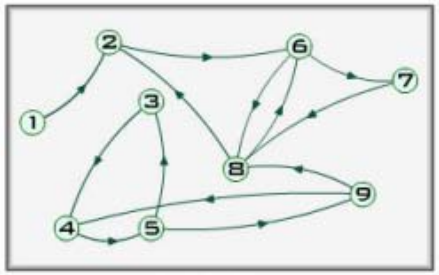
Un graf orientat G=(X, U) este **tare conex**, dacă pentru oricare două noduri x și y ∈ X, există un drum de la x la y precum și un drum de la y la x.

EXEMPLU

Graful G_1 este tare conex iar graful G_2 nu este tare conex. G_2 are 3 **componente tare conexe**.



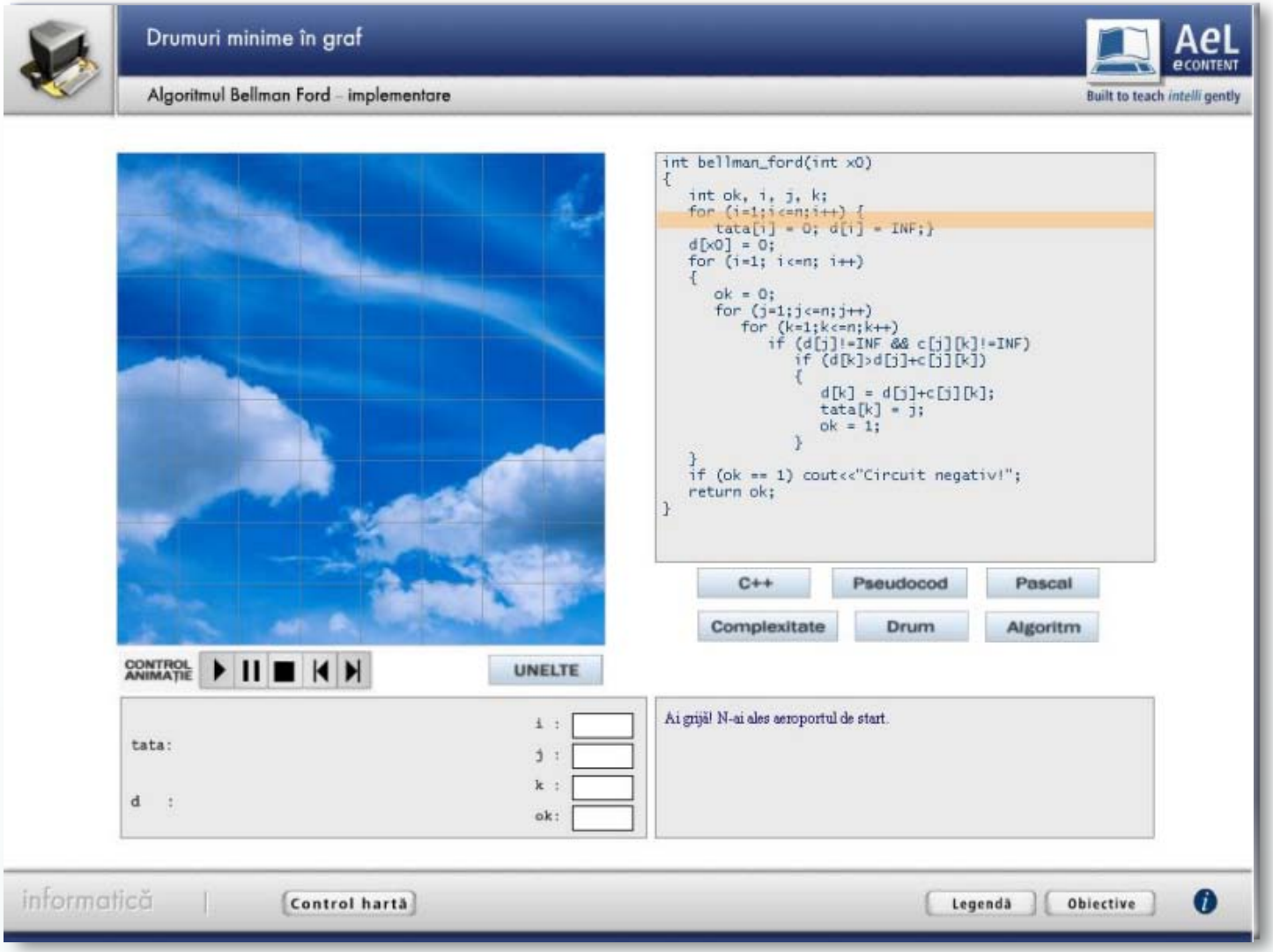
Graful G_1



Graful G_2

SIVECO ROMANIA SA

Drumuri minime în graf. Partea III

- **Recomandat pentru doua ore de predare.**

1. Algoritmul lui Belman Ford: Prezentarea problemei
2. Algoritmul lui Belman Ford: Reprezentarea informatiilor
3. Algoritmul lui Belman Ford: Implementare
4. Algoritmul lui Belman Ford: Evaluare